Univerzita Palackého v Olomouci

Přírodovědecká fakulta

Katedra geoinformatiky

Studijní program: **P1314 Geografie**

Obor: **1302V011-02 Geoinformatika a kartografie**

DISTRIBUOVANÉ GEODATABÁZE SENZOROVÝCH DAT

ZÁKLAD PRO INTEGRACI A ANALÝZU

Doktorská disertační práce

Mgr. Tomáš POHANKA

Vedoucí práce: doc. RNDr. Vilém PECHANEC, Ph.D.

Olomouc 2020

ANOTACE

Disertační práce se věnuje analýze, návrhu a ověření architektury systému, umožňující sběr, uložení, zpracování a sdílení velkého množství dat generovaných moderními geograficky rozmístěnými senzorovými sítěmi s využitím distribuovaných prostorových databázových systémů s následným využitím těchto dat v geografických informačních systémech pro tvorbu časoprostorových analýz. Práce předkládá autorský přístup k budování senzorových sítí pomocí LPWAN, předzpracování těchto dat pro využití v GIS a vytváření distribuované databázové sítě pro zvýšení dostupnosti dat. Motivací k disertační práci byla potřeba zpřístupnit proudová senzorová data uživatelům GIS tak, aby bylo možné jejich pokročilé zpracování pomocí časoprostorových analýz v reálném čase. Práce odpovídá ve své experimentální části na dvě základní výzkumné otázky (hypotézy):

- Jsou v současnosti dostupné replikační mechanismy v prostorových databázích použitelné pro zabezpečení distribuce prostorových a senzorových dat?
- Je efektivní provést integraci heterogenních senzorových dat do jednotného datového úložiště?

Pro potvrzení či zamítnutí výše uvedených hypotéz byl hlavní cíl práce rozdělen na tři dílčí cíle. První dílčí cíl měl popsat a otestovat dostupná replikační řešení v návaznosti na jejich funkčnost s prostorovými daty. Druhý dílčí cíl se zabýval zpracováním senzorových dat produkovaných senzorovými sítěmi komunikujících pomocí technologií Sigfox, LoRaWAN, Zigbee a GPRS. Třetí dílčí cíl si kladl za úkol získaná data očistit a uložit do jednotné prostorové databáze senzorových dat. Databatáze byla dále distribuována pomocí replikačních mechanismů tak, aby se zamezilo centralizaci dat.

Klíčová slova replikace; prostorová databáze; senzor; komunikace; middleware

Počet normostran textu: 103

ANNOTATION

The main objective of this doctoral thesis was analysing and verification of system architecture, which enable to collect, store, process and share large-scale data generated by modern, geographically distributed wireless sensor networks. Architecture is using distributed spatial database systems for creation of spatiotemporal analysis.

Thesis present author's approach for building wireless sensor networks using LPWAN (Low Power Wide Area Networks) technologies, preprocessing of obtained data for future usage in GIS and creating distributed spatial database site to increase data availability. The motivation for the doctoral thesis was a demand to make streaming sensor data accessible in desktop GIS in real-time. Desktop GIS is mostly used for creating advanced spatiotemporal analysis.

The thesis answers two primary research questions:

1. Are there nowadays available replication mechanisms for spatial databases suitable for distribution of spatial and sensor data?

2. Is it efficient to integrate heterogeneous sensor data into unified data storage?

The main objective of doctoral thesis was split into three sub-objectives in order to confirm or reject the hypothesis. The first sub-objective aims to describe and test available database replication solutions which provide replication functionality over spatial data. The second sub-objective aims to processing of sensor data produced by LPWAN sensor networks Sigfox, LoRaWAN, Zigbee and GPRS (not low powered). The third sub-objective aims to purge obtained data from gross errors and storing it into the unified spatial database. Database was distributed by replication mechanisms to avoid data centralisation.

Keywords replication; spatial database; sensor; communication; middleware

Number of standard pages: 103

Prohlašuji, že

- disertační práci včetně příloh, jsem vypracoval samostatně a uvedl jsem všechny použité podklady a literaturu,
- jsem si vědom, že na moji disertační práci se plně vztahuje zákon č.121/2000 Sb. - autorský zákon, zejména § 35 – využití díla v rámci občanských a náboženských obřadů, v rámci školních představení a využití díla školního a § 60 – školní dílo,
- beru na vědomí, že Univerzita Palackého v Olomouci (dále UP Olomouc) má právo nevýdělečně, ke své vnitřní potřebě, disertační práci užívat (§ 35 odst. 3),
- souhlasím, aby jeden výtisk disertační práce byl uložen v Knihovně UP k prezenčnímu nahlédnutí,
- souhlasím, že údaje o mé disertační práci budou zveřejněny ve Studijním informačním systému UP,
- v případě zájmu UP Olomouc uzavřu licenční smlouvu s oprávněním užít výsledky a výstupy mé disertační práce v rozsahu § 12 odst. 4 autorského zákona,
- použít výsledky a výstupy mé disertační práce nebo poskytnout licenci k jejímu využití mohu jen se souhlasem UP Olomouc, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly UP Olomouc na vytvoření díla vynaloženy (až do jejich skutečné výše).

V Olomouci dne Mgr. Tomáš POHANKA

Rád bych poděkoval všem, kteří mi v průběhu tvorby disertační práce poskytli pomoc. Především panu docentu RNDr. Vilému Pechancovi, Ph.D., za trpělivost, velmi cenné rady, věcné připomínky a vstřícnost při konzultacích. Jeho skvělé vedení velmi přispívalo realizaci této disertační práce po celou dobu studia.

Rád bych poděkoval i celému kolektivu Katedry geoinformatiky, protože každý člen přispěl k pohodové i produktivní atmosféře na pracovišti. Zejména pak RNDr. Janu Brusovi, Ph.D. a doc. RNDr. Jaroslavu Burianovi, Ph.D. za jejich užitečné rady v průběhu celého studia. Děkuji také současným i bývalým doktorandům, se kterými jsem spolupracoval na projektech, ale i na každodenním doktorandském životě. Zvláště pak Mgr. Tomáši Pourovi, Ph.D. za jeho energický přístup k řešení problémů.

Velmi děkuji celé rodině, která mi byla vždy oporou. Obzvlášť pak Pavlovi, který jako první přišel s myšlenkou studia na olomoucké geoinformatice a se kterým jsem prokonzultoval mnoho nocí. Poděkování patří i sestře Petře, matce Jitce i zesnulému otci Zdeňkovi, kteří vždy za mnou stáli a podporovali mě.

Mimořádný dík patří i mé přítelkyni Tereze za její trpělivost a podporu během psaní disertační práce.

Obsah

Přehled zkratek

ACID	Atomic, Consistent, Isolated, Durable
ANSI	American National Standards Institute
ArcSDE	Spatial Database Engine
ASCII	American Standard Code for Information Interchange
AWS	Amazon Web Services
B	byte
b	bit
BLOB	Binary Large Object
BPEJ	Bonitovaná půdně ekologická jednotka
BPSK	Binary-Phase Shift Keying
BSON	Binary JavaScript Object Notation
BTS	Base Transciever Station
CMS	Content Management System
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
CRa	České Radiotelekomunikace
CSV	Comma Separated Values
DBMS	Database Management System
DDBS	Distribuovaný Databázový Systém
DDS	Data Distribution Service
DPDBS	Distribuovaný Prostorový Databázový System
ESB	Enterprise Service Bus
FDW	Foreign Data Wrappers
FTP	File Transfer Protocol
GB	Gigabyte

Gbps	Gigabit per second
GDAL	Geospatial Data Abstraction Library
GIS	Geographic Information System
GNU	GNU's Not Unix
GPL	GNU General Public License
GPRS	General Packet Radio Service
GRDDL	Gleaning Resource Descriptions from Dialects of Languages
GSM	Groupe Spécial Mobile
HSPA	High Speed Packet Access
HTTP	Hypertext Transfer Protocol
IAB	Internet Architecture Board
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IRI	Internationalized Resource Identifier
ISO	International Organization for Standardization
jpeg	Joint Photographic Experts Group
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
KGI	Katedra geoinformatiky
LOB	Large Object Binary
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LPN	Low-Power Network
LPPAN	Low Power Personal Area Network
LPWAN	Low Power Wide Area Network
LTE	Long Term Evolution

LZW	Lempel-Ziv-Welch
MB	Megabyte
Mbps	Megabit per second
MQTT	Message Queuing Telemetry Transport
MVCC	Multiversion concurrency control
NB-IoT	Narrow Band – Internet of Things
NDB	Network DataBase
NFC	Near field communication
NFS	Network File System
NoSQL	Not Only SQL
O&M	Observations and Measurements Schema
OData	ISO Open Data Protocol
ODMG	Object Data Management Group
OGC	Open Geospatial Consortium
OGR	Geospatial Data Abstraction Library
OMG	Object Management Group
OOP	Oběktově orientované programování
OQL	Object Query Language
ORDBMS	Object-relational Database Management System
ORM	Object-relational mapping
OSWA	Open Sensor Web Architecture
OWL	Web Ontology Language
P2P	Peer-2-Peer
PaaS	Platform as a Service
PAN	Personal Area Network
PHP	Hypertext Preprocessor

png	Portable Network Graphics
POWDER	Protocol for Web Description Resources
QoS	Quality of Service
QPSK	Quadrature phase-shift keying
R2RML	RDB to RDF Mapping Language
RAM	Random Access Memory
Rasdaman	Raster data manager
rasql	Rasdaman Query Language
RBR	Row-Based Replication
RDB2RDF	Relational Databases to RDF
RDBMS	Relation Database Management System
RDF	Resource Desciption Framework
RDFS	RDF Schema
REST	Representational State Transfer Application Programming Interface
RIF	Rule Interchange Format
RLE	Run-length encoding
RSS	Recieved Signal Strength
RTC	Real-time clock
RTOS	Real-time operating system
SaaS	Software as a Service
SAS	Sensor Alert Service
SAWSDL	Semantic Annotations for WSDL and XML Schema
SBR	Statement-Based Replication
SD	Secure Digital
SDWWG	Spatial Data on the Web Working Group
SensroML	Sensor Model Language

SEQUEL	Structured English Query Languag
SHACL	SHapes And Constraints Language
SKOS	Simple Knowledge Organization System
SMS	Short Message Service
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOS	Sensor Observations Service
SOSA	Sensor, Observation, Sample. and Actuator
SPARQL	SPARQL Protocol and RDF Query Language
SPOF	single point of failure
SPS	Sensor Planning Service
SQL	Structured Query Language
SSB	Star Schema Benchmark
SSL	Secure Sockets Layer
SSN	Semantic Sensor Network Ontology
SWE	Sensor Web Enablement
TAČR	Technologická agentura České republiky
TB	Terabyte
TML	Transducer Markup Language
TPC-x	Trasaction Processing Performance Council
URI	Uniform Resource Identifier
UTF	UCS/Unicode Transformation Format
UTP	Unshielded Twisted Pair
WCPS	Web Coverage Processing Services
WCS	Web Coverage Service
WKB	Well-Known Binary

WKT	Well-Known Text
WNS	Web Notification Service
WoT	Web of Things
WSDL	Web Service Description Language
XAMPP	Cross-Platform, Apache, MariaDB, PHP a Perl
XHTML	Extensible Hypertext Markup Language
XML	eXtensible Markup Language
YAML	YAML Ain't Markup Language
YCSB	Yahoo Cloud Services Benchmark
ZB	Zettabyte

Seznam obrázků

Seznam tabulek

1 ÚVOD

Data jsou nedílnou součástí geografických informačních systémů spolu se softwarem, hardwarem a pracovníky, kteří se systémem pracují a využívají ho. Žádná z těchto součástí nesmí chybět, ovšem důležitost jednotlivých součástí se liší. Hardware v dnešní době není nákladný a specializovaný hardware není často nezbytný. Uživatel má možnost výběru software od komerčních poskytovatelů i jako open-source či freeware. Další součástí pro geografické informační systémy nezbytné, jsou data. Data mohou být buď zakoupena, využít otevřená data, nebo data získat z vlastních informačních zdrojů, např. terénním sběrem dat, snímky pořízené dronem či automatizovaný sběr dat pomocí senzorů a senzorových sítí [93].

Dle predikce od roku 2015 do roku 2025 vzroste dvojnásobně (z 15 % na 30 %) počet strojově generovaných dat a jejich objem bude dosahovat téměř 80 ZB *(zettabyte)* [128, 137]. I každodenní interakce s digitálním světem se s novými technologiemi (mobilní telefony, hodinky, náramky) výrazně zintenzivňuje a roste produkce nových strojových dat. Úloha přenést tato data do GIS prostředí je nezpochybnitelná. Produkování vlastních senzorových dat je jen první fáze. V dalších fázích je třeba data zpracovávat, často v reálném čase, ukládat, analyzovat a výsledky z těchto analýz publikovat. Pro prezentaci senzorových dat je nutné zohlednit aspekty (požadavky na) zpracování dat v senzorových sítích, databázových systémech a i v GIS.

Při integraci heterogenních senzorových sítí do centralizovaného databázového úložiště mohou vznikat problémy spojené s celkovým výkonem zařízení. Jedním z možných řešení je centralizované databázové úložiště fyzicky rozdělit, ovšem při zachování logické centralizace (z pohledu uživatele). Vytvořením distribuované databázové sítě je zajištěna robustnost celého systému, která inteligentně řeší rozložení zátěže, dostupnost a ochranu dat.

2 CÍL PRÁCE

Cílem této disertační práce je analýza, návrh a ověření architektury systému, umožňující sběr, uložení, zpracování a sdílení velkého množství dat generovaných moderními geograficky rozmístěnými senzorovými sítěmi s využitím distribuovaných prostorových databázových systémů s následným využitím těchto dat v geografických informačních systémech pro tvorbu časoprostorových analýz.

Disertační práce se snaží zodpovědět dvě základní výzkumné otázky (hypotézy):

- Jsou v současnosti dostupné replikační mechanismy v prostorových databázích použitelné pro zabezpečení distribuce prostorových a senzorových dat?
- Je efektivní provést integraci heterogenních senzorových dat do jednotného datového úložiště?

Pro naplnění cíle disertační práce jsou výzkumné aktivity rozděleny na tři logické, na sebe navazujících dílčí cíle. Výsledky a závěry z dílčích cílů v souhrnné podobě slouží pro potvrzení či zamítnutí základních hypotéz. Práce není ovlivněna kvalitou vstupních dat, která velmi souvisí s cenou jednotlivých čidel a účelu nasazení senzorových sítí.

Dílčí cíl 1 – Replikační mechanismy databázových systémů

Prvním dílčím cílem je popis a testování distribuovaných databázových systémů (*DDBS*), jejich vlastností, parametrů a funkcionalit. Definování typů distribuovaných databází a uvedení rozdílů mezi DDBS a distribuovanými prostorovými databázovými systémy (*DPDBS*), včetně podrobného popisu jejich možných přístupů a řešení je nezbytným předpokladem. Jsou uvažovány relační databáze i databáze NoSQL (*Not Only SQL*). Hlavní důraz je kladen na DPDBS. Primárním předmětem studia distribuovaných (prostorových) databází je forma a způsob využívání replikačních a synchronizačních procesů prostorově orientovaných dat, které jsou zde otestovány a popsány, včetně im-

plementačních nároků jednotlivých řešení DPDBS. Spolu s popisem replikačních a synchronizačních procesů je popsán a otestován celkový přínos nasazení distribuovaných databázových systémů na fungování a management prostorových dat. Na dílčích případových studiích jsou shrnuty základní poznatky z implementací a jejich dopady na fungování celé distribuované databázové sítě. Výsledkem dílčího cíle 1 je doporučené nasazení konkrétních distribuovaných databázových systémů a dále jsou nastíněny případy jejich využívání. Uvedená doporučení jsou založena na kombinaci teoretických znalostí vycházejících z rešeršního rozboru a praktických znalostí získaných v průběhu vlastního testování.

Dílčí cíl 2 – Ověřený postup zpracování dat ze senzorové sítě

Druhý dílčí cíl se zabývá zpracováním dat ze senzorů a senzorových sítí, které mají potenciál stát se jedním z hlavních producentů primárních dat [62, 128]. V rešerši jsou zahrnuty aspekty konceptu Internetu věcí (IoT), který je dnes jedním z nejrychleji se rozvíjejících technologických oborů. Jako hlavní zdroj senzorových dat pro potřeby výzkumu jsou použity senzory monitorující fyzicko-geografické vlastnosti krajiny. Dále se rozbor věnuje moderním přenosovým technologiím z oblasti *Low Power Wide Area Network* ($LPWAN$) pro IoT. V České republice jsou momentálně nabízeny dvě celorepublikové technologie Sigfox a LoRaWAN a nově také NB-IoT a LTE-M. U těchto technologií jsou popsány jejich technologické, ale i implementační aspekty. V průběhu práce byla navržena a otestována architektura senzorové sítě komunikující prostřednictvím technologie LoRa, LoRaWAN a Sigfox. V neposlední řadě jsou diskutovány nové přístupy k získávání polohy bez potřeby globálních družicových polohových systémů, který nyní nabízí technologie Sigfox a v budoucnu LoRaWAN.

Dílčí cíl 3 – Integrace senzorových dat do jednotné prostorové databáze

Třetí dílčí cíl se zabývá integrací dat spojenou s přenosem dat ze senzorů do jednotné prostorové databáze. Hlavní důraz je kladen na formy, stupně a možnosti integrace dat do jednotné prostorové databáze. Jsou popsány přístupy přenosu dat v reálném čase (*real-time*), proudová data a jejich využití. Pro příjem dat je naprogramován integrační mechanismus, který integruje data přijímaná pomocí technologií LoRaWAN, LoRa, Sigfox, Zigbee a GPRS (*General Packet Radio Service*) do jednotné prostorové databáze. Mechanismus mimo jiné řeší chyby v senzorových datech, například omezení rozsahu hodnot poskytovaných senzory. Dále jsou popsány možné struktury prostorové databáze, na jejímž základě je navrhnuta struktura prostorové databáze tak, aby byla využitelná jak pro geografické informační programy, tak pro aplikační rozhraní. Navržená struktura databáze je následně použita do navržené distribuované databázové sítě.

Jako příklad aplikačního využití dosažených výsledků je uvedena architektura pro integraci, přenos, uložení a následnou analýzu v produktu QGIS. Ten v současnosti jako jediný GIS umožňuje využít funkci *Notify* z PostgreSQL.

3 METODY A POSTUP ZPRACOVÁNÍ

Cíle práce je dosaženo splněním dílčích cílů a potvrzením či zamítnutím hypotéz. V rámci prvního dílčího cíle bylo provedeno testování za účelem určení vhodného způsobu replikace prostorových dat. Druhý dílčí cíl řešil zpracování dat ze senzorových sítí a popisuje komunikace mezi senzorovými sítěmi a senzorovou prostorovou databází. V rámci třetího dílčího cíle byla zpracovaná data integrována do jednotné prostorové databáze, která je zároveň distribuována tak, aby se zamezilo datové centralizaci. Následně byl zajištěn přenos (primárních či analyzovaných) dat z distribuované databázové sítě do GIS pomocí konceptu *Publisher − Subscriber*. Získané znalosti a podklady vedly k vytvoření architektury celého řešení pro přenos, uložení, zpracování a využití prostorových dat z různých zdrojů v rámci desktop GIS pro možnost provádění časoprostorových analýz.

3.1 Replikační mechanismy databázových systémů

V rámci řešení prvního dílčího cíle byla provedena rešerše z literárních zdrojů týkajících se především prostorových databází a distribuovaných prostorových databází. Základní vlastnosti, výhody a nevýhody jednotlivých komerčních i open-source řešení jsou popsány v podobě textu a tabulek. Při návrhu ukazatelů vhodnosti použití a využití distribuované prostorové databáze je čerpáno z nejnovější literatury a aktuálních verzí databázových systémů tak, aby byla zajištěna co nejvyšší aktuálnost nabízených funkcionalit. Aktuální funkcionalita z pohledu databázových i integračních možností často převyšuje funkcionalitu samotných softwarových řešení geografických informačních systémů. Uvažované distribuované prostorové databázové systémy jsou z oblasti relačních i z oblasti NoSQL systémů. Výběru jednotlivých databázových systémů předcházela rešerše o možnostech integrace databázových systémů a geografických informačních systémů a jejich využitelnosti při návrhu distribuované prostorové databázové sítě. Rešeršní část obsahuje popis testování, včetně odlišností mezi jednotlivými systémy, zejména pak jakým způsobem

pracují s replikačními a synchronizačními mechanismy v distribuované databázové síti. Odlišnost mechanismů (replikačních a synchronizačních) nespočívá jen v různorodém pojmenování, ale i v logice fungování a nabízených architekturách.

V praktické části tohoto dílčího cíle bylo prováděno testování a hodnocení replikace prostorových dat. V rámci experimentu byly testovány dva databázové servery – PostgreSQL 9.5 s PostGIS 2.3.3, Slony 2.2.8 a MySQL 5.7.19. PostgreSQL verze 9.5 byl testová z důvodu využívání této verze na katederním serveru. Aktuálně se jedná o nejnižší stále podporovanou verzi (až do roku 2021). Ostatní podporované verze jsou 9.6, 10, 11 a 12.

PostgreSQL databáze obsahovala rozšíření PostGIS a replikační nástroj Slony-I, který vytváří replikační logiku z hlavní databáze do podřízené databáze (*master – slave replication*). Do *master* databáze lze vkládat nově přijatá data. Do *slave* databáze může zapisovat pouze *master* databáze. *Slave* databáze je určena jen ke čtení, mimo zápis nových dat z *master* databáze. U MySQL databáze bylo využito vestavěné replikační funkce. Testování probíhalo pod mým vedením v rámci bakalářské práce Bc. Lenky Trnové [146]. Tyto systémy byly používané na Katedře geoinformatiky (*KGI*) v době testování a byly nasazeny na serveru katedry. Výběr databázových systémů PostgreSQL a MySQL podléhalo několika kritériím (tabulka 1).

Tabulka 1: Kritéria výběru databázového systému

Kritérium	PostgreSQL	MySQL (MariaDB)	Oracle	MS SQL	MongoDB
Využitelnost na KGI	ano	ano	ne	ne	ne
Multiplatformní	ano	ano	ano	ano	ano
Hardwarové požadavky	minimální	minimální	vyšší	vyšší	minimální
Softwarové požadavky	minimální	minimální	vyšší	vyšší	minimální
Uložení a zpracování prostorových dat	ano	ano	ano	ano	omezeně
Napojení na GIS software	ano	ano	ano	ano	omezeně
Uživatelská základna	celosvětová	celosvětová	celosvětová	celosvětová	celosvětová

Testování probíhalo na datových sadách, které jsou běžně dostupné a používané na území České republiky. Jedná se o datové sady ArcČR500 v3.3 (geografická a topografická data pro měřítko 1 : 500 000), Data200 (data pro mapová měřítka 1 : 200 000 vycházející z EuroRegionalMap), NaturalEarth v3.0.1 (data pro mapová měřítka 1 : 10 000 000 obsahující kulturní a fyzické vrstvy i rastrová data pro celý svět) a BPEJ (bonitované půdně-ekologické jednotky) v5.1.2018 pro Olomoucký kraj. Bližší specifikace použitých dat lze nalézt v tabulce 2. Pro uložení prostorových dat byl použit standard WKB (*Well-Known Binary*) definovaný v OGC (*Open Geospatial Consortium*) Simple Feature for SQL (*Structured Query Language*) 1.2.1. Zásadní hodnoty pro replikace jsou počet lomových bodů a počet záznamů. Lomový bod u bodové vrstvy je totožný s bodem.

Tabulka 2: Popisné informace použitých dat pro testování replikačních mechanismů

Název vrstvy	Zdrojová sada	Datový typ	Lomové body [počet]	Záznamy [počet]	Velikost [MB]
BPEJ	Státní pozemkový úřad	Polygon	3 725 023	31 280	228
Části obce	ArcČR500	Body	15 092	15 092	12
Řeky	Data200	Linie	338 959	14 606	21
Města	Data200	Polygon	672 299	6 353	27
Svět	NaturalEarth	Polygon	411 132	1	4

Byla ověřena rychlost samotné synchronizace a identifikována „úzká hrdla“ (*bottleneck*) přenosu dat. Aby byla zajištěna co nejvyšší reálnost testování byly databázové systémy nainstalovány na dvou separátních zařízeních. Pokud by byly servery nainstalovány na jednom zařízení, eliminoval by se vliv linkové vrstvy, v tomto případě vliv UTP (*Unshielded Twisted Pair*) patch kabelu. Jako zařízení pro testování byl použit standardní osobní počítač a notebook. Detailní popis sestav je uveden v tabulce 3.

Tabulka 3: Hardwarová konfigurace serverů

Parametr	Osobní počítač	Notebook
Operační systém	Microsoft Windows 8.1	Microsoft Windows 10
RAM	8 GB	4 GB
CPU	Intel Core i5-4590	Intel Core i3-2330M
Grafická karta	Nvidia GeForece GTX-960	Nvidia Geforce GT-540M
Síťová karta	Realtek RTL8110G	Broadcom NetLink

Testování probíhalo ve třech režimech propojení serverů:

1. propojení zařízení přes 100 Mbps (*megabit per second*) router,
2. propojení zařízení přes router s omezením šířky pásma na 10 Mbps,
3. přímé propojení zařízení přes 100 Mbps UTP patch kabel kategorie 5E.

Replikační proces byl spouštěn v konfiguraci *master* − *slave* a test probíhal v konfiguraci, kdy *master* server byl počítač a *slave* server byl notebook, tak i v konfiguraci obrácené, tedy *master* server byl notebook a *slave* server byl počítač. Vzniklo tedy celkem šest různých konfigurací pro každý databázový server, při kterých byla testována robustnost a výkon replikačních mechanismů při manipulaci s prostorovými daty.

Pro hodnocení replikačního klastru byl sledován celkový čas replikace a vytíženost CPU (*Central Processing Unit*) a sítě. Replikační klastr je prostředí, ve kterém jsou databáze propojené replikačním mechanismem. Jsou tedy fyzicky oddělené ale logicky propojené. Úspěšnost replikace prostorových dat byla 100 %, proto nebyla úspěšnost replikace brána jako hodnotící kritérium. Testování obsahovalo dvě základní operace s prostorovými daty. První operací byla změna geometrie dat a druhou byla změna atributové hodnoty. Každé měření probíhalo desetkrát z důvodů eliminace okolních vlivů. Více než deset měření již nevykazovalo žádnou variabilitu celkového času replikace, rychlosti přenášených dat či využití procesoru. První úloha (editace geometrických údajů prostorových dat) byla provedena v softwaru QGIS 2.18.10, který umožňuje přímé napojení na PostgreSQL a MySQL databáze. Editace probíhala tak, že byly označeny všechny body a následně došlo k náhodnému geometrickém posunutí. Tím se změnila geometrie všech lomových bodů. Pokud by se editoval jen jeden lomový bod, replikace by proběhla tak rychle, že by ji nebylo možné zachytit měřícími programy. Proto byly vrstvy zvoleny tak, aby měly různý, avšak vyšší počet lomových bodů. Druhá úloha (editace atributových údajů prostorových dat) byla provedena přímo SQL dotazem v *master* serveru. Aby bylo možné replikaci změřit, bylo najednou změněno 30 000 údajů v atributu.

3.2 Ověřený postup zpracování dat ze senzorových sítí

Druhý dílčí cíl se zabývá přenosovými technologiemi z oblasti LPWAN, které jsou vhodné pro prostředí IoT a jsou využitelné v České republice a ve světě, včetně jejich přínosu při sběru a zpracování senzorových proudových dat pro GIS.

Pro realizaci experimentů (této části) byly využity senzorové systémy, kterými disponuje Katedra geoinformatiky. Jedná se zejména o patnáct senzorů firmy Libelium. Pro platformu Libelium (základem je Arduino) byly v roce 2017 zakoupeny anténní moduly pro komunikační technologie Sigfox a LoRaWAN. Ty byly vybrány v závislosti na dostupnosti technologie v České republice a pro podporu projektů TAČR (Technologická agentura České republiky) Bezkontaktní monitorování a časoprostorové modelování variability vybraných diferenciačních vlastností půdy (TA04020888) a Systém automatizovaného monitorování a modelování znečištění podzemních vod z nebodových průmyslových zdrojů (TH03030023). Čidla pro snímání environmentálních veličin byla použita ta již zakoupená spolu se senzory Libelium, i čidla zakoupená od jiných výrobců. Jako například srážkoměry, čidla pro měření půdní vlhkosti nebo čidla pro měření elektrické vodivosti vody. Kvalita naměřených veličin a s tím související následná interpretace naměřených dat není v rámci disertační práce řešena. Kvalita naměřených veličin je velmi úzce spjata s cenou čidla a tím i s účelem použití [47].

U technologií Sigfox a LoRaWAN jsou popsány jejich registrační podmínky a způsob připojení do celorepublikové (světové) sítě. Popis je vytvářen ve spolupráci s poskytovateli SimpleCell, respektive Českými Radiotelekomunikacemi (CRa). Dále je u technologie Sigfox diskutována technologie geolokace senzoru bez potřeby globálních družicových polohových systémů. Tato funkcionalita je založena na vyhodnocování úrovně přijatého signálu komunikačními bránami. Možnosti využití této služby a přesnost geolokace jsou prakticky ověřeny.

Výsledkem druhého dílčího cíle je analýza, návrh a ověřená architektura heterogenních senzorových sítí, které produkují dynamická proudová data (více o proudových datech v kapitolách 4.3.2 na straně 54 a 4.5 na straně 72). Senzory byly programovány v nativních programovacích jazycích pro danou platformu. Libelium Waspmote je programován v jazyce C++. Čidly, kterými disponuje Libelium je možné měřit vzdušnou teplotu a vlhkost, CO_2 a organické těkavé látky. K Libelium je také možné připojit meteorologickou stanici s anemometrem, srážkoměrem a větrnou směrovkou. Zapojení čidel jiných výrobců do senzorů Libelium vyžadovalo fyzické úpravy vnitřního zapojení. Senzory Libelium nabízí šest pozic pro zapojení jejich čidel. Každé čidlo může být využito jen na určité pozici [92]. Dále byly nutné úpravy i v samotných knihovnách, které jsou kompilovány spolu s kódem přímo do paměti základní desky senzoru. Základ všech senzorů Libelium tvoří jednočipový počítač ATmega 1281. Příkladem úpravy knihovny je například změna z výpočtu objemu srážek na prostý čítač impulzů, aby bylo možné připojení obecného srážkoměru s překlopným člunkem s laboratorně změřeným objemem na jedno překlopení. Další úpravy zahrnovaly například rozšíření knihovny o možnost měření elektrické vodivosti vody. Celkem vznikly tři prototypy senzorů Libelium, které měří vzdušnou teplotu a vlhkost čidlem Sensiron SHT75, 2× půdní vlhkost čidlem Decagon EC5, elektrickou vodivost DFRobot DFR0300, teplotu vody pro kalibraci elektrické vodivosti teplotním čidlem DS18B20 a srážkoměr Amat (více v tabulce 7 na straně 91). Dva senzory komunikují na technologii Sigfox (mohou i na LoRaWAN) a třetí na technologii Zigbee. Časový interval odesílání dat byl nastaven na 15 minut. Senzory Libelium používají komunikační protokoly Zigbee (dostupných 15 antén), Sigfox (dostupné tři antény) a LoRaWAN (dostupné tři antény). Dále jsou využity dataloggery firmy Ekotechnika, které komunikují na technologii GPRS. U LPWAN bylo zapotřebí uzpůsobit velikost a frekvenci odesílaných dat. Velikost dat byla omezena na 12 B dat v jedné zprávě. Každá odesílaná hodnota byla převedena na hexadecimální hodnoty. Naměřené

hodnoty byly upraveny tak, aby jejich rozsah byl mezi celočíselnými hodnotami 0 − 255 (více v kapitole 5.2 na straně 91). U dataloggeru je výstupní formát neměnný.

Dále byly naprogramovány koncové body komunikačních technologií Sigfox Cloud a IoT Portal pro LoRaWAN tak, aby bylo možné pomocí naprogramovaného integračního mechanismu zapisovat data do jednotné prostorové senzorové databáze.

3.3 Integrace senzorových dat do jednotné prostorové databáze

Při řešení třetího dílčího cíle se vycházelo z potřeby integrace heterogenních senzorových sytému do jednotné databáze tak, aby byla tato proudová senzorová data možné dále využívat v GIS systémech. Provedení integrace je nutná pro budoucí využití senzorových dat v desktop GIS, jelikož heterogenita není jen ve vlastnostech senzoru, ale i v datech, která lze ze senzorů získat (z výsledků v kapitole 5.2 na straně 91). Způsobená centralizace senzorových dat v jednotné prostorové databázi je vyřešena metodami vytvářející distribuované prostředí, zejména replikačními mechanismy (z výsledků v kapitole 5.1 na straně 74).

Každá komunikační technologie (Zigbee, Sigfox, LoRaWAN, LoRa) poskytuje data v různých formátech (JSON, prostý text, SQL, hexadecimální kód) a v různých objemech (např. 1 b − 12 B pro Sigfox technologii). Naprogramovaný integrační mechanismus, který automaticky rozpozná přijatá data, je součástí webového serveru, který obsluhuje příjem dat, řeší hrubé chyby senzorových dat (hodnoty mimo rozsah měření) a provádí ukládání dat do jednotné prostorové databáze. Webový server i integrační mechanismus je napsán v programovacím jazyce Python. Webový server poskytuje REST API (*Representational State Transfer Application Programming Interface*), které zabezpečuje příjem dat od poskytovatelů (Sigfox a LoRaWAN). České Radiotelekomunikace požadují webový server s SSL (*Secure Sockets Layer*) zabezpečením. Certifikát lze zakoupit a využít server Katedry geoinformatiky, nebo využít služeb komerčních poskytovatelů cloudových slu-

žeb, například Microsoft Azure. Proudová senzorová data posílaná přes Zigbee a zachytávaná branou Libelium Meshlium jsou uchovávána v MySQL přímo v bráně, následně jsou replikována na externí server, kde jsou uložena v MariaDB. Přímé propojení MySQL a PostgreSQL umožňují různá řešení, ať již například FDW *(Foreign Data Wrappers)* či SymmetricDS. Ovšem, s přihlédnutím na dnešní běžnou praxi uchovávání veškerých senzorových data v jediné tabulce, byl nejlépe použitelný replikační systém pg_chameleon. Tento replikační nástroj je napsán v jazyce Python 3 a jen pro systémy Linux. Vytváří permanentní replikační spojení mezi MySQL (MariaDB) a PostgreSQL. Nástroj je konfigurovatelný přes YAML (*YAML Ain't Markup Language*) soubory [45].

Přijatá zpráva od Sigfox či LoRaWAN senzorů obsahuje nejen data poslané senzorem (např. teplota, vlhkost, tlak), ale i data o síle signálu, kolik bran zprávu přijalo, čas přijetí zprávy bránou, lokalizaci brány. Sigfox nabízí možnost uživatelsky nadefinovat odchozí zprávy. České Radiotelekomunikace mají jednotný výstupní formát.

Návrh struktury databáze vychází z rešeršní části prvního dílčího cíle. Při návrhu schématu je počítáno s implementací do distribuované prostorové databázové sítě rámci relačních databází (PostgreSQL).

Návrh struktury databáze vycházel z rešeršní části prvního dílčího cíle, ze standardu OGC SensorThing API, z open-source IoT platformy ThingsBoard a z produkční databáze brány Meshlium firmy Libelium. Statická část databáze, která se nemění či jen velmi zřídka, obsahuje metadata o senzorech, čidlech, ale i poloze samotného senzoru. Tyto informace mohou být obsáhlé a rozdělené do jednotlivých tabulek. Návrh dynamické části databáze, která obsahuje výsledky měření čidel senzorů, může být pojata z různých pohledů:

1. každé čidlo každého senzoru má vlastní tabulku,
2. každý senzor má vlastní tabulku se všemi čidly,
3. jediná tabulka pro všechny čidla všech senzorů.

Každý návrh má své výhody i nevýhody. Jedním faktorem je složitost SQL dotazu na jeden senzor, jedno čidlo, jednu lokalitu, kde řešení 1 bude mít výhodu oproti řešení 3. Opak představuje situaci, kdy je požadováno vyhledání všech čidel nebo senzorů v čase či lokalitě. Dalším faktorem je velikost tabulky a s tím související doba vyhledání konkrétních záznamů. Jak řešení ThingsBoard, tak Meshlium využívají 3. metodu, tedy veškerá naměřená data od všech senzorů v jednotné univerzální tabulce.

Pro praktické ověření celé architektury distribuované prostorové databáze senzorových dat, byl vytvořen její návrh a byla ověřena její funkčnost. Byl využit desktop GIS QGIS 3.10, který od verze 3 pracuje s PostgreSQL funkcí *Notify/Listen*, která funguje na konceptu *Publisher – Subscriber*. PostgreSQL na základě definované funkce a spouště (*trigger*) odesílá notifikaci o vzniklé události všem připojeným uživatelům, kteří sledují daný odběr. Podobnou technologii využívají i další databázové systémy, např. CouchDB či MongoDB.

4 TEORETICKÁ VÝCHODISKA A SOUČASNÝ STAV

Současná infrastruktura prostorových informací budovaná od 90. let 20. století je založena na poskytování dat pouze na žádost (dotaz/odpověď, *request/response*). Stejně tak současný desktop GIS software (ArcGIS, Geomedia, TerrSet, GRASS GIS, QGIS, SAGA GIS, JUMP GIS, gvSIG ...) téměř ve všech případech tuto funkcionalitu podporuje a dokáže s touto logikou pracovat. Historicky byla data velmi těžko dohledatelná, nebyla on-line, byla v proprietárním datovém formátu a chyběla metadata [49]. Dnes již díky vytvořeným standardům a technologickému pokroku není velkou překážkou ani jedno.

Soudobý technologický vývoj však směřuje k aplikacím, které vyžadují reakci na události, jež se staly, byly zjištěny, změřeny, zaznamenány a s minimálním časovým zpožděním poskytnuty. Jedná se především o oblast krizového managementu, precizního zemědělství, předpovědních služeb a informování o aktuálním stavu. Díky velmi rychlému vývoji bezdrátových komunikačních technologií, robustnějších, levnějších a dostupnějších senzorů lze budovat ohromné senzorové sítě o tisících senzorových jednotkách. Vyšší dostupnost představuje příležitost pro rozšíření osobních senzorů (mobilní telefon, domácí meteorologická stanice), které produkují obrovské množství heterogenních dat. A právě díky bezdrátovým technologiím lze tato data zpřístupnit na Internet velmi rychle a bez velké energetické náročnosti.

Senzorová data mění podobu zpracování dat současnými desktop GIS, které jsou stále nenahraditelné v celém procesu získávání, uložení zpracování a publikování dat. Desktop GIS by měly začít podporovat nové metody jak přistupovat a přijímat data, inovovat kódování a výměny časoprostorových dat z velmi jednoduchých zařízeních (např. v rámci IoT konceptu) a v neposlední řadě přijmout a podporovat nové standardy pro výměnu dat [129].

Oproti architektuře dotaz/odpověď přináší událostmi řízená architektura (*event-driven architecture*) do budoucna podstatné vylepšení celého procesu zpracování proudových prostorových dat. Dnes již existují tradiční technologie, které lze použít pro zpracování proudových dat. Jsou jimi například Apache Storm, Apache Kafka, Apache Flink, Apache Spark. Vylepšený desktop GIS může mít následující strukturu [129]:

1. Přijetí zprávy o specifické události (např. překročení limitu).
2. Včasné doručení prostorových dat v okamžiku jejich zpřístupnění (např. nový satelitní snímek).
3. Omezení síťové komunikace (již nebude potřeba odesílat dotaz, jestli jsou nová data).
4. Spouštění procesní události v okamžiku přijatých nových prostorových dat.

Integrace proudových prostorových dat v desktop GIS v rámci událostmi řízené architektury stále chybí. Pro web a WebGIS, který je mnohem flexibilnější a rychlejší v nasazování nových technologií, je práce s proudovými daty jednodušší, a již existují platformy, které proudová data podporují, například ArcGIS GeoEvent Server.

4.1 Databázové systémy

Tato kapitola se věnuje popisu vybraných druhů databázových systémů, které byly uvažovány pro splnění cílů dizertační práce.

4.1.1 Počátek databází a objektově orientovaných databází

Na začátku 70. let 20. století publikoval Edgar Frank Codd, tehdejší vědec u společnosti IBM, článek *A Relational Model of Data for Large Shared Data Banks* (Relační datový model pro rozsáhlé sdílené databáze). Codd [52] ve svém článku poprvé definoval vlastnosti relačních databází.

Codd vydal v roce 1971 publikaci *Data Base Sublanguage Founded on the Relational Calculus* (Tvorba specializovaného databázového jazyka pro práci s relacemi). Článek pracoval s Coddem vytvořeným jazykem Alpha, který byl nakonec i základ jazyka SQL. SQL jazyk je specializovaný jazyk vytvořený pro řízení dat v relačním databázovém systému. Podle návrhu E. F. Codda, i když ne úplně podle jeho představ (bez jeho účasti a bez jazyka Alpha), byl návrh v roce 1973 IBM implementován do databázového systému – *System R*. Následný vývoj SQL jazyka probíhal pod dohledem Donalda D. Chamberlina a Raymonda F. Boyce, v té době znám jako SEQUEL (*Structured English Query Language*) pro Systém R. Boyce spolu s Coddem paralelně pracovali na Boyce-Codd normálních formách pro efektivní návrh tabulek pro relační databázi. Po téměř šesti letech vydaly firmy Oracle (Oracle Version 2) a IBM (SQL/DS) první komerční databáze postavené na relačním systému využívající SQL jazyk [107, 142]. V roce 1983 byla vydána první verze databáze DB2 od IBM. I v dnešní době, po více než 40 letech od první zmínky o jazyce SQL a relačních databázích, se tato kombinace stále těší velké oblíbenosti [18]. SQL jazyk byl v roce 1986 Americkým národním institutem pro standardy (*ANSI – American National Standards Institute*) standardizován. O rok později byl přijat i Mezinárodní organizací pro standardy (*ISO – International Organization for Standardization*). První standardizovaná verze SQL jazyka byla SQL 86, nebo zjednodušeně SQL1. Další větší revize jazyka byly SQL2 a SQL3, které byly publikovány v roce 1992, respektive v 1999.

V roce 2003 byla do SQL přidána podpora XML (*eXtensible Markup Language*), velkých objektů (*LOB – Large Object Binary*), které mohou mít až 128 TB (terabyte). V pořadí pátá verze byla vydána v prosinci 2016 (ISO/IEC 9075-1:2016).

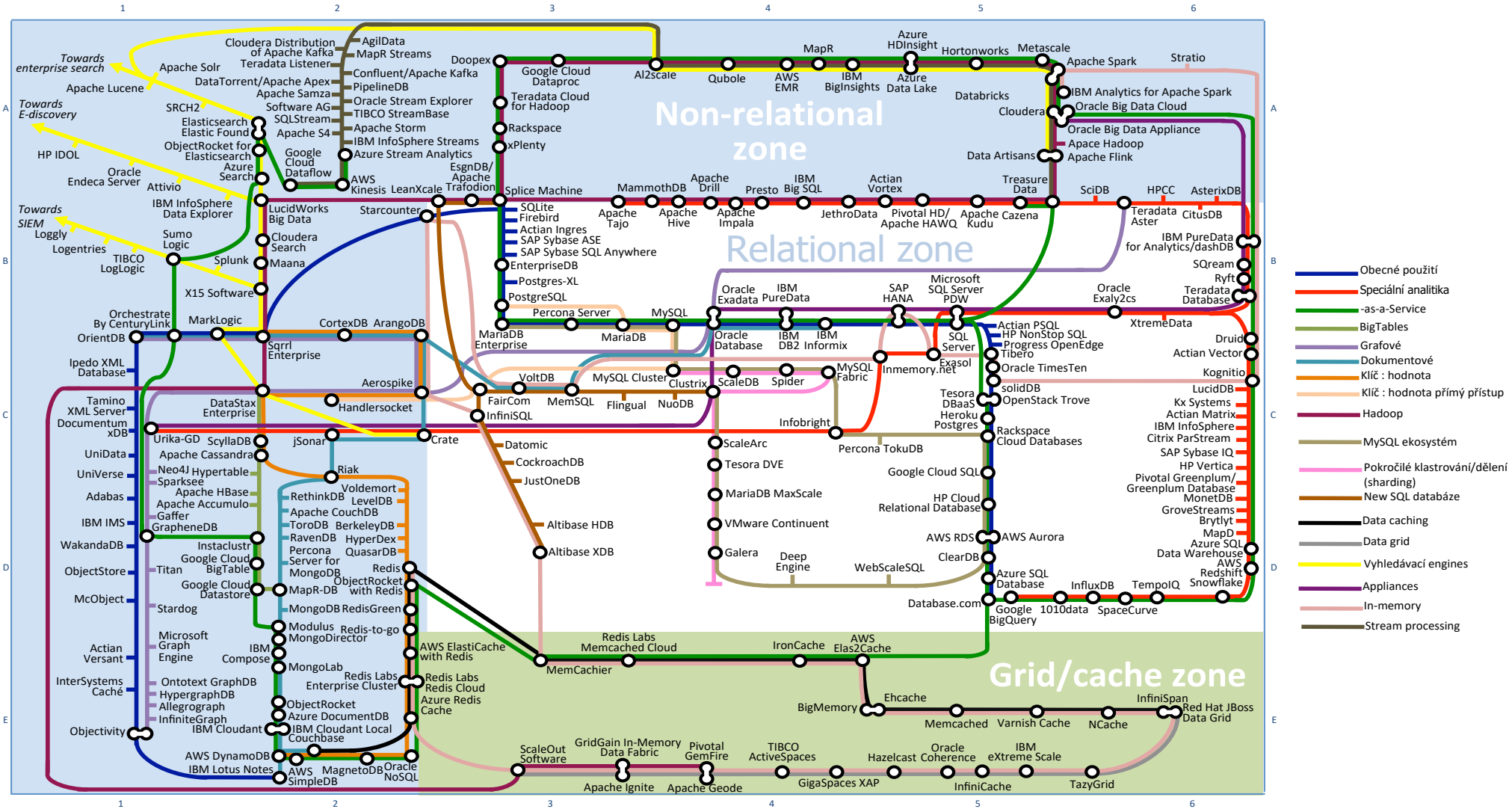
Relační databázový model pro správu řízení báze dat (*DBMS – Database management system*) není jediným modelem, který v současné době existuje. Mezi další modely lze zařadit například flat (Berkley DB, TextDB, Mimesis, MySQL CSV), hie-

rarchický (doplněk k DB2 od IBM), síťový (TurboIMAGE, RMD), objektový (db4o), objektově-relační (PostgreSQL, MySQL, MariaDB), klíč-hodnota (Redis), dokumentový (MongoDB, Couchbase), multihodnotový (Rasdaman, SciDB) apod. Podle DB-engines.com [18] existuje celkem 14 různých modelů ukládání dat v celkem 351 různých databázových systémech (obrázek 1). Pro komunikaci uživatel – databáze jsou vytvořeny DBMS, které slouží pro správu dat a zabezpečují komunikaci mezi uživatelem a daty uloženými v binární podobně na pevném disku. Uživatel komunikuje s relačním DBMS (*RDBMS*) pomocí SQL jazyka, jež DBMS přeloží do strojového kódu a vyžaduje po disku data, který je vrátí v binární podobě, jež DBMS opět přeloží do uživatelem čitelného jazyka podle určitého kódování (nejčastěji *UTF-8*).

Objektové databáze byly tvořeny od 80. letech 20. století. Motivace pro jejich vytvoření bylo v tehdejší době využívané objektové programování. Existovala snaha sjednotit programovací jazyk a uložená data tak, aby oboje pracovalo s objekty bez potřeby konverzních nástrojů mezi relační databází a objektovým programováním. Objekt uložený v databázi poté dědil třídy, vlastnosti i metody. Objekt je vždy spravován jako celek. Například, při vkládání objektu, který by byl v relačním modelu zřejmě rozdělen do několika tabulek, je zpracován automaticky jako jedna atomická transakce. Čtení objektu je zpracováno jako jeden příkaz, bez potřeby komplexních spojení tabulek [23]. Základní vlastností objektově orientovaných databází je čisté propojení s objektově orientovaným programováním (*OOP*) (C++, Java, Python a další). Pro OOP jsou charakteristické čtyři vlastnosti: dědičnost, datové zapouzdření, objekty a polymorfismus. Jelikož existuje velké zastoupení relačních databázových systémů a jsou nejběžnější variantou při ukládání dat, musí se pro OOP (pro objekty a třídy) mapovat na databázové tabulky definované v relačním schématu. Tato programovací technika je označována jako ORM (objektově relační zobrazení, *object-relational mapping*). ORM zajišťuje automatickou konverzi dat mezi relační databází a OOP. Entita (reprezentace reálného světa)

je v relační databázi reprezentována jako řádek či skupina řádků v databázových tabulkách a v objektově orientovaném jazyce je entita zpravidla reprezentována jako instance třídy. Někdy je velmi obtížné zachytit entity reálného světa do řádků a tabulek v relační databázi. S tím souvisí zpětné získávání informací z relační databáze. Mezi relačními a objektovými databázemi je přechodná skupina takzvaných objektově-relačních databází (*ORDBMS – object-relational database management system*). Hlavní představitel ORDBMS je PostgreSQL. Oracle ve svých databázích nabízí i objektový přístup, ovšem nemá kompletní implementaci. Například nelze vytvořit vlastní datový typ (jedním ze základů objektového programování). Jelikož objektově orientovaný model a relační model má mnoho odlišných předpokladů, je velmi obtížné oba přístupy naplno propojit. Proto je vhodnější označovat ORDBMS jako relační databázový model s objektově orientovaným rozšířením.

Skupina ODMG [48] (*Object Data Management Group*), která se starala o chod objektově orientovaných databází, byla založena na jaře 1991. Hlavní osobou byl Rick Cattell z firmy Sun Microsystems. Hlavním cílem skupiny ODMG bylo vytvářet přenositelné specifikace pro vývojáře aplikací pro objektově orientované databáze a objektově relační mapovací nástroje. Mezi lety 1993 a 2001 vydala skupina pět revizí specifikace. Po poslední specifikaci ODMG ver. 3 (v roce 2001) byla skupina rozpuštěna. Hlavní komponenty specObject Data Management Groupifikace ODMG ver. 3 jsou 1) objektový model; 2) objektově orientovaný jazyk, postaven nad objektovým modelem; 3) objektový dotazovací jazyk (*OQL − Object Query Language*); 4) vazba k C++ jazyku; 5) vazba k Smalltalk jazyku; 6) vazba k Java jazyku. V roce 2004 byla nová skupina OMG *(Object Management Group*) pověřena revizí specifikace ODMG ver. 3 pod právy Morgan Kaufmann Publishers. Na začátku roku 2006 zveřejnila skupina OMG skupinu ODBT WG (*Object Database Technology Working Group*) a plány na čtvrtou generaci specifikace pro objektově orientované databáze.



Obrázek 1: Schéma databázových platforem. Zdroj: [9]

4.1.2 Relační DBMS

Tato kapitola se zabývá popisem vybraných relačních databázi. Relační databáze jsou databáze, které využívají pro uložení dat tabulky, které jsou vzájemně logicky propojeny. Díky svému výkonu, standardizaci a rozšíření se dnes jedná o nejvíce používaný typ databází v GIS.

4.1.2.1 PostgreSQL

Začátek open-source projektu PostgreSQL je na Kalifornské Univerzitě v Barkeley [15]. V roce 1996 se Postgres95 (verze z roku 1995) odtrhl od akademické sféry a vstoupil do open-source světa a byl vyvíjen mimo Univerzitu Barkeley. Během dalších osmi let vytvořili dobrovolníci konzistentní a sjednocený kód, vytvořili podrobné testy pro měření kvality, dále systém pro hlášení a opravu chyb, přidali nespočet nových vylepšení a funkcionalit a doplnili dokumentaci pro vývojáře a uživatele. Jejich úsilím byla vytvořena nová databáze, která má silnou reputaci a stojí na pevných základech. Díky novým funkcím, stabilnímu jádru a open-source licenci se Postgres95 přejmenoval na PostgreSQL, i když Postgres je stále hojně využívané pojmenování.

PostgreSQL je multiplatformní a běží na všech hlavních systémech jako jsou Linux, Unix (AIX, BSD, SGI IRIX, Mac OS X, Solaris, Tru64) a Windows. Pracuje výhradně v ACID (*Atomic, Consistent, Isolated, Durable*) módu. ACID je zkratka pro atomicitu, konzistenci, izolaci a trvalost. Atomicita zajišťuje, že se databázová transakce provede celá nebo vůbec. Konzistence zajišťuje, že transakce neporuší žádné integritní omezení. Izolovanost zajišťuje, že operace uvnitř transakce jsou nezávislé na vnějších operacích (především pro funkci *rollback*, která vrací stav databáze před provedením operace). Trvalost zajišťuje, že úspěšně provedené operace jsou trvale zapsány v databázi a nemohou být ztraceny [15]. PostgreSQL má několik velikostních limitů:

Tabulka 4: Velikostní omezení PostgreSQL databáze v12. Zdroj: [15]

Parametr	Limit
Maximální velikost databáze	Neomezeno
Maximální velikost tabulky (blok)	32 TB (8kB) – 128 TB (32kB)
Maximální velikost pole	1 GB
Maximální počet řádku v tabulce	Neomezeno
Maximální počet atributů v tabulce	250 – 1600 (podle datového typu)
Maximální počet indexu v tabulce	Neomezeno

PostgreSQL podporuje procedurální jazyky jako PL/PGSQL nebo Perl bez nutnosti nahrávat externí moduly. Objektově orientované jazyky Python či Java jsou dostupné pomocí modulů. Prostředí PostgreSQL dovolí vytvoření rozhraní pro správu a transformaci dat. PostgreSQL využívá pro ukládání dat pouze relační model, ovšem dovoluje uložení neatomických datových typů jako jsou pole, XML (*eXtensible Markup Language*), JSON a hstore.

4.1.2.2 MySQL

MySQL databáze patřila švédské firmě MySQL AB (Michael Widenius, David Axmark a Allan Larsson) od roku 1995, která databázi vyvíjela přímo jako open-source [13]. V roce 2008 firmu koupila firma Sun Microsystems, Inc., za jednu miliardu dolarů. O dva roky později koupila firmu Sun Microsystem firma Oracle za 7,4 miliardy dolarů. Pro Oracle to znamenalo nejen získání konkurenčního databázového systému, ale i vlastnictví technologií jako je Java či NFS (*Network File System*). Oracle se nákupem firmy Sun Microsystem stala vedle softwarové firmy i hardwarovou [136]. To jí umožnilo nabízet komplexní služby a řešení využívající vlastní hardware a software. Oracle začal nabízet MySQL jako open-source komunitní verzi pod licencí GNU (*GNU's Not Unix*) GPL (*GNU General Public License*), i komerční verzi, která nabízí rozšířené možnosti správy, propojení a zabezpečení. Po akvizici firmou Oracle vytvořil Michael Widenius open-source větev MariaDB, která je plně kompatibilní s MySQL.

MySQL je multiplatformní databázový server, který je velmi populární pro webové servery jako datové úložiště. Využívají ho přední CMS (*Content Management System*) software pro správu webového obsahu jako jsou například WordPress, Joomla! a Drupal. V GIS světě není ovšem MySQL tak rozšířen, i když práci s prostorovými daty podporuje. MySQL implementuje OGC specifikace OpenGIS® Implementation Standard for Geographic information – Simple feature access – Part 2: SQL option i SQL/MM Part 3: Spatial. MySQL nachází přímou podporu v rámci implementace knihovny GDAL/OGR (*Geospatial Data Abstraction Library/OpenGIS Simple Features Reference Implementation*). Od MySQL verze 8.0.0 (k dispozici vývojářům od 12. 6. 2016, stabilní verze vydána 19. 4. 2018) změnila práci s prostorovými daty, respektive přepsala názvy funkcí, které nekorespondovaly se standardem, například: *AsText() -> ST_AsText()*. Verze 8 přinesla velké zlepšení podpory GIS a velmi se přibližuje rozšíření PostGIS pro PostgreSQL.

4.1.3 NoSQL

NoSQL (*Not Only SQL*) nikdy neměly být náhradou relačních databází. Termín NoSQL se může přeložit jako databáze, kde dotazovací jazyk nemusí být jen standardní SQL. NoSQL databáze používají nejrůznější datové modely [18]. Mezi nejčastější používané datové modely patří dokumentové, grafové, klíč-hodnota, objektové, vyhledávací, širokosloupcové, RDF, časových řad, multidimenzionální.

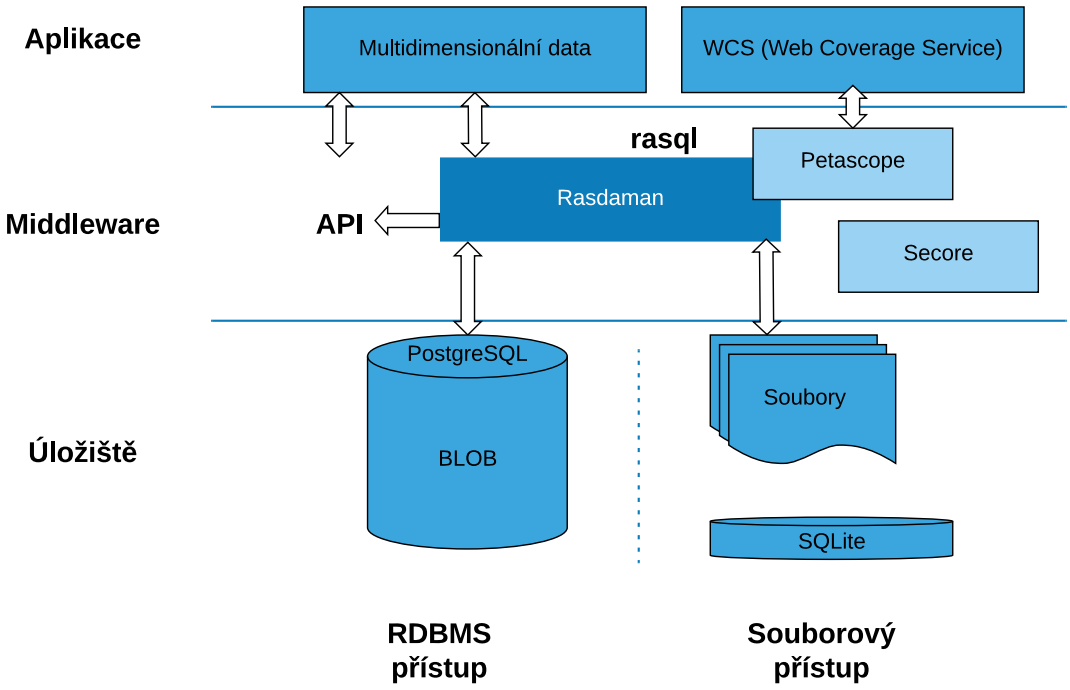
Spolu s nejnovějšími verzemi hlavních databázových producentů se ovšem setkáváme s implementací pro podporu většího množství datových modelů v jednom databázovém systému. Například Oracle nabízí kromě obligátního relačního modelu i dokumentový, grafový a RDF model. PostgreSQL a MySQL nabízí také dokumentový model.Nejpopulárnější NoSQL databáze jsou MongoDB, Elasticsearch, Redis a Cassandra [18]. Pro dokumentový datový model v zásadě platí, že jsou data uložena jako JSON,

případně BSON, který v roce 2009 představil MongoDB. Výhoda ukládání dat v JSON dokumentu je v absenci pevného schématu. Každý dokument tedy nemusí obsahovat všechny „položky“.

V GIS světě NoSQL databáze upevňují svou pozici [2, 40, 66, 161, 1, 30, 133, 123, 158]. Seznam všech podporovaných prostorových databází jistě nemůže být finální, ale podle možností knihovny GDAL/OGR, kterou používá drtivá většina GIS software, lze najít například MongoDB, CouchDB, Cloudant (IBM implementace CouchDB), ElasticSearch, TileDB [155]. Další NoSQL databáze mohou implementovat vlastní prostorové funkce, jako například Redis [127]. Esri v produktech ArcGIS (Desktop i Pro) používá některé funkce z knihovny GDAL/OGR, ale konektivitu pro NoSQL nenabízí. QGIS (verze 3.10) je na tom o mnoho lépe a nabízí ve formě plug-in propojení na výše uvedené NoSQL databáze. Další velmi používaná NoSQL databáze pro Web GIS je CouchDB, která skvěle pracuje s JavaScript, a je zaměřena především na webovou vizualizaci. NoSQL dokumentové databáze (CouchDB, MongoDB) využívají k uložení vektorových dat JSON, respektive GeoJSON struktury. Pro rastrová data nabízí MongoDB GridFS. Pro jakékoli JSON dokumenty větší než 16 MB jsou data uložena právě v GridFS. CouchDB pracuje s velkými soubory jako s přílohami, které jsou uloženy v nativní podobě na disku. Je to výhoda například pro **.jpeg* či **.png* rastrové dlaždice.

Pro práci s rastry je od roku 1989 vyvíjen middleware Rasdaman (*Rastr data manager*). Byl jedním z prvních, který se zaměřoval speciálně na rastrová data, ale i na jakákoli data v polích (*array database*). Rastrová data jsou uložena přímo v nativním souborovém formátu nebo v relační databázi (PostgreSQL). Souborový systém je často výhodnější a efektivnější než uložení jako BLOB (*Binary Large Object*) v databázi [160]. Metadata jsou uložena v SQLite databázi (při souborovém uložení dat), nebo přímo v relační databázi. Rasdaman se chová jako middleware, který přináší vlastní dotazovací jazyk rasql (*Rasdaman Query Language*) pro manipulaci s daty. Dotazy v rasql jsou

poté Rasdaman parsovány a vykonávány s tím, že jsou data přijata z relační databáze či z jednotlivých souborů. Rasdaman nabízí také aplikaci Petascope, která implementuje některé OGC webové služby, například WCS (*Web Coverage Service*) či WCPS (*Web Coverage Processing Services*). Z multidimenzionálních databází (*multidimensional array database*) mohou těžit např. producenti na poli satelitního monitorování Země (systémy Landsat či Sentinel), kteří produkují velké množství rastrových dat v časových řadách. Z těch je pak možné vytvořit datové kostky a spouštět pokročilé časoprostorové analýzy.



Obrázek 2: Architektura Rasdaman. Zdroj: [160]

I když PostgreSQL obecně nemá problémy s nízkým výkonem, v porovnání s MongoDB má při multiuživatelském přístupu s požadavkem na vložení či čtení časové problémy. MongoDB dosáhlo při multiuživatelském přístupu (1 000 uživatelů) a požadavku na čtení 3,48 % času PostgreSQL [50]. V další studii byla zkoumána možnost zpracování videa s využitím GIS a i v této studii dopadla relační databáze zastoupená MySQL hůře, než NoSQL databáze MongoDB [158]. Neznamená to ovšem, že NoSQL

databáze jsou obecně vhodnější pro využití v GIS. Jeden z problémů NoSQL je neschopnost provést spojení dat (*join*). Pokud je potřeba sloučit určitá data, nelze to provést „virtuálně“ spojením, ale data se musí vytvořit. Tím vzniká velké množství duplicitních informací a počet dokumentů. Informace by jiným způsobem nebylo možné získat. Při srovnání výkonosti na jeden dotaz od jednoho klienta, tak PostgreSQL s PostGIS odpovídá mnohem rychleji [96]. Při použití streaming replikace a middleware PGPool v klastru 5 instancí, jak pro PostgreSQL, tak pro MongoDB, je rozdíl ještě zřetelnější.

NoSQL databáze nejsou dokonalé a s relačními databázemi se nemohou zaměňovat. Analytické schopnosti NoSQL databází nad prostorovými daty jsou minimální. MongoDB umožňuje tři prostorové dotazy: a) geoIntersects (v PostGIS *ST_Intersects*), b) geoWithin (*ST_Within*), c) near (*ST_DWithin*). Další analytické zpracování je nutné řešit v rámci aplikace. Pro složitější analytické či komplexnější dotazy a propojení s GIS software, je nejvhodnější zůstat u relační prostorové databáze [35].

4.1.4 Prostorové databáze

Prostorová databáze je jakýkoli databázový systém, který umožňuje uložení prostorových dat ve speciálním datovém typu pro prostorová data (podle ISO a OGC v datovém typu geometrie či geografie). Geodatabáze je datový formát vyvinutý společností Esri. Všechny geodatabáze jsou prostorové databáze, ale ne všechny prostorové databáze mohou být geodatabází.

Firma Esri zavedla pojem geodatabáze poprvé ve svém software ArcGIS 8 v roce 1999 jako proprietární datový formát pro práci s ArcGIS. Geodatabáze může být vytvořena v produktech Esri, nebo může být zprostředkovaná ArcSDE (*Spatial Database Engine*) middleware technologií, která umožňuje práci s relačními databázemi Oracle, Microsoft SQL Server, IBM Informix, IBM DB2 a PostgreSQL (pro verzi ArcGIS 10.7.1). ArcSDE middleware je součástí ArcGIS Server v balíčku ArcGIS Enterprise.

Od verze ArcGIS Desktop 10 (2010) vznikl pro GDAL ovladač pro čtení i zápis do geodatabáze FileGDB, který vychází z FileGDB API od Esri. API je od roku 2016 vedeno pod licencí Apache Licence 2.0. Další možností je využít OpenFileGDB ovladač, který byl vytvořen zpětným inženýrstvím ze souborové geodatabáze. Oba přístupy ovšem nejsou bezchybné, OpenFileGDB je pouze pro čtení, FileGDB (v 1.5.1) má problémy s vytvořením a zápisem do geodatabází vytvořené v ArcGIS Pro 2.4 [155]. I přes tyto problémy se stala geodatabáze přístupná i pro širokou platformu open-source softwarů.

Prostorová data jsou data o poloze, tvaru a vztazích mezi jevy reálného světa, vyjádřená zpravidla ve formě souřadnic a topologie [26]. Prostorová data mohou být v databázi uložena jako prostý text, nejčastěji ve formátu WKT (*Well-known Text*). WKT je definován jako standard ISO/IEC 13249-3:2016 s původním návrhem specifikace OpenGIS® Implementation Standard for Geographic information – Simple feature access – Part 1: Common architecture [105]. Specifikace definuje jednotlivé typy geometrie a základní dotazy a analýzy nad geometrií. I když mohou být prostorová data uložena v databázi jako text, je pro snadnější využití dat v GIS vhodnější jejich uložení do prostorové databáze, která rozšiřuje databázi o další funkcionalitu. Nejčastěji prostorové databáze implementují OpenGIS Simple feature specifikaci. Prostorová data jsou poté uložena v prostorové databázi v datovém typu geometrie (*geometry*), která pracuje se souřadnicemi v kartézské projekci či typu geografie (*geography*), která pracuje se souřadnicemi na kouli.

4.1.5 Distribuované databázové technologie

Distribuovaný databázový systém spojuje technologie databázových systémů a technologie počítačových sítí [108]. Hlavní motivací, proč využít databázové systémy je integrace využívaných dat do jednoho místa (centralizace) s následnou kontrolou nad tímto místem. Naopak, počítačová síť jde opačným směrem a snaží se nabídnout decentralizovaný

přístup. Hlavní otázkou tedy zůstává, jak tyto dvě technologie mohou vůbec kooperovat. Hlavní myšlenkou distribuovaných databázových systémů tedy není jejich centralizace, ale integrace dat bez jejich centralizace [108]. Distribuovaná databáze se může definovat jako kolekce logicky propojených a fyzicky rozdělených databází v počítačové síti [108, 87]. Lze jí docílit databázovými replikacemi, horizontálním či vertikálním dělením nebo kombinací uvedeného [110]. Spojením více databázových systémů jsou vytvořeny distribuované databázové sítě, které zamezují centralizaci, a tím neúměrnému zatížení jednoho místa v síti.

Jeden z prvních distribuovaných databázových systémů byl budován v první polovině 70. let 20. století australským ministerstvem obrany [87]. Ministerstvo mělo centrální databázový systém, ovšem jeho části byly replikovány na základě lokálních potřeb po celé Austrálii. Systém nesynchronizoval repliky okamžitě, ale synchronizace probíhala několikrát denně. Dnes již není problém udržovat distribuované systémy na síti Internet či intranet, které komunikují nepřetržitě. Jako hlavní důvody, proč distribuovat části nebo celou databázi uvádí [87]:

- Efektivní uložení dat, tam kde jsou využívána a tím zamezení nadměrného přenosu dat na velké vzdálenosti.
- Ochrana dat pomocí částečné replikace tak, že organizaci nejsou poskytnuta data, se kterými nepracují.
- Využívání zdrojů, kde není potřeba spravovat celé tabulky s daty, ale pouze jejich náhledy s agregovanými daty.
- Využití paralelismu, kde jsou podobné transakce zpracovány paralelně v různých sítích.

Distribuovaný databázový systém nepřináší jen výhody, má i nevýhody:

- Systém musí být rozšířen o správu distribuovaného prostředí.
- Špatné rozdělení dat může vést k horším výkonům databáze (z důvodu vysoké komunikace mezi uzly), což může vést k problémům s integritou dat.
- Musí být zajištěn robustnější transakční systém, který řídí transakce mezi uzly.

Distribuované databáze jsou vytvářeny typicky databázovými replikacemi [108, 126]. Další technikou může být shrading. Při shradingu si každý databázový uzel (dva a více) drží svou část dat. Všechny uzly dohromady poté tvoří celou datovou sadu [87]. Čtyři hlavní účely databázových replikací jsou:

1. Zabezpečení přístupnosti systému – pokud selže jeden uzel, je uživatel automaticky přesměrován na další. Funkcionalitu rozdělování zdrojů mohou mít na starosti PGPool či MySQL Router. Odstraní se tím také kritické místo poruchy (*single point of failure – SPOF*).
2. Výkon – nejen rozdělování uživatelů na méně vytížené databázové servery v distribuovaném databázovém klastru, ale i skladování dat v místě vyššího využívání či kratší vzdálenosti od odběratelů.
3. Rozšiřitelnost – rozšiřování počtu dostupných uzlů přidáním dalšího databázového serveru do klastru.
4. Aplikační potřeby – replikace může být vyžadována aplikací, která si udržuje mnoho kopií dat jako část její funkčnosti.

4.2 Synchronizace a replikace

Odlišení pojmů replikace a synchronizace může být obtížné. Obecně je databázová replikace technologie či proces pro kopírování a distribuování dat a databázových objektů z jedné databáze do druhé. Replikace může probíhat v rámci jednoho databázového serveru na jednom místě, nebo mezi vzdálenými databázovými servery v různých lokalitách [5]. Následně probíhá synchronizace mezi těmito databázemi k zajištění konzistentnosti a integrity dat [79]. Replikační technologie využívají synchronizační přístupy, jako například jednocestnou či obousměrnou synchronizaci [97]. Před budováním distribuovaného prostředí je nezbytné porozumět datům a aplikaci [150].

Databázovou replikací je míněn proces sdílení informací k zajištění konzistence mezi redundantními zdroji (databázemi), zlepšující spolehlivost a přístupnost a snižující chybovost [99]. Databázová replikace vytváří kopie dat, která jsou pak přístupná z různých míst (jiný server). Replikační technologie poté hlídá změny provedené v databázi a synchronizuje změny mezi databáze v distribuovaném prostředí [150]. Uživatelské dotazy mohou být poté směrovány na různé, méně vytížené databázové servery, nebo se více databázových serverů může podílet na zpracování výsledku. Databázová replikace by se neměla zaměňovat s datovým skladištěm (*data warehouse*), které slučuje různá datová úložiště, s rozdílným schématem [131, 76, 108]. Databázové replikace vytváří kopie celé databáze, vybraných tabulek či vybraných dat na další databázové servery. Replikace vytváří mezi databázovými servery spojení a zabezpečuje jednosměrnou (*one-way*) či obousměrnou (*two-way*) komunikaci [108, 121]. Volba replikace databází by se měla zvážit především u sdílení dat mezi vzdálenými pracovišti, při využívání dat uživateli z celého světa, zvýšení přístupnosti serveru a zabezpečení zálohy dat. Replikace by ovšem neměla být jediná forma zálohy dat a nemělo by se k replikacím přistupovat jako k čistě zálohovacímu mechanismu. Replikaci podporují relační databázové systémy i NoSQL. Oba typy databázových systémů jsou vhodné k uchování prostorových informací,

a je na konkrétní aplikaci, zda dokáže vytěžit z databázových technologií jejich přednosti [133]. Primárním cílem databázových replikací je vytvářet redundantní, propojené distribuované prostředí.

Termíny distribuovaný databázový systémem a databázová replikace spolu úzce souvisí, ale nesmí se zaměňovat. U distribuovaného systému se vyskytuje pouze jediná kopie dat a typicky je využívaná u distribuovaných transakcí, které využívají lokální i vzdálenou databázi k vykonání požadavků v reálném čase [144]. Naopak, databázová replikace pracuje s více shodnými databázemi, které jsou propojeny uvnitř distribuované databázové sítě. Tím umožňuje lokálním aplikacím pracovat s lokální databází, která má nejaktuálnější globální data [58].

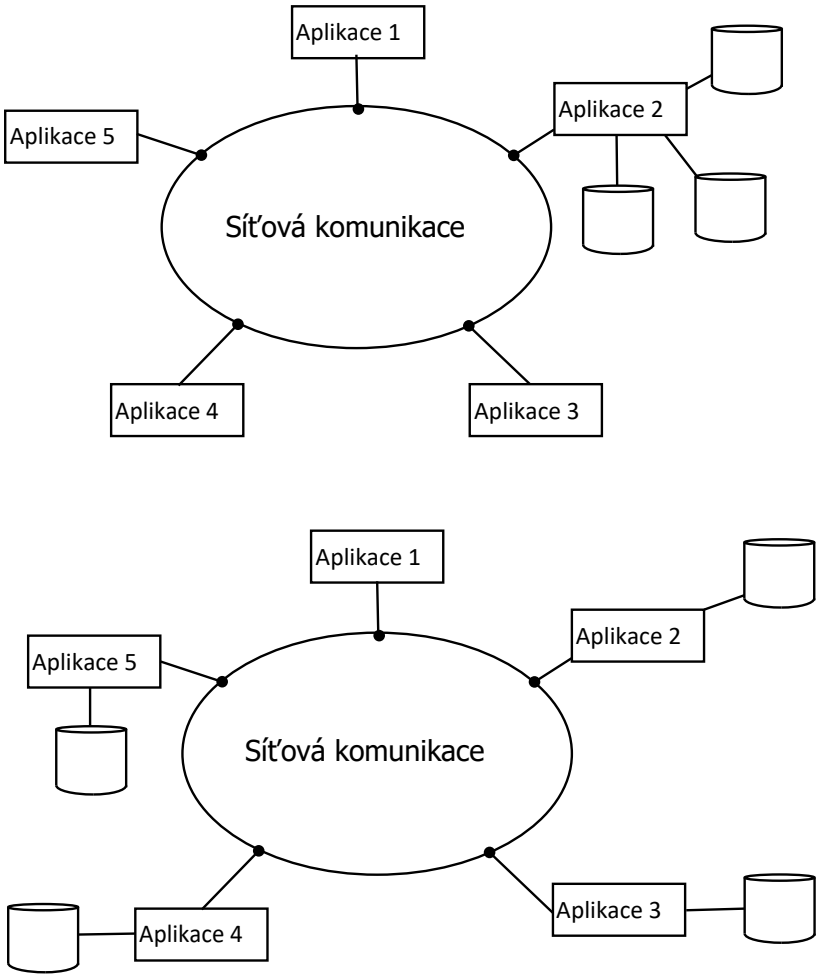
Základním stavebním prvkem databázové replikace je uzel (*node*), který reprezentuje jeden databázový server či klastr. Uzel může mít roli nadřízenou (*master*) serveru či podřízenou (*slave*) serveru. Je běžnou praxí používat spojení *master* server a *slave* server v replikačním prostředí. Tyto názvy nejsou standardizované a každá společnost nabízející replikační mechanismy může použít jakýkoliv termín. Mohou to být například názvy pro *master* server primární (*primary*), vydavatel (*publisher*), vedoucí (*leader*). Pro *slave* servery se používají názvy pohotovostní či záložní (*standby*), odběratel (*subscriber*), následník (*follower*). Názvy mohou mít spojitost s určitým typem replikace, technologií či společností, nicméně často označují totožný replikační mechanismus [3]. V replikačním prostředí s minimálním počtem dvou serverů v konfiguraci *master–slave* (jednocestná) replikace má *master* server roli zapisovatele a příjemce zpráv. Jeho hlavní úlohou jsou dotazy na vložení (*insert*) a aktualizaci (*update*) záznamů. *Slave* server poté slouží čistě pro čtení dat (*select*). Při konfiguraci *multimaster* (obousměrná) replikaci jsou obě (při konfiguraci dvou serverů) databáze schopné zapisovat a aktualizovat data [34]. Při *multimaster* replikaci musí být ošetřeno řešení konfliktů (např. *MVCC − Multiversion concurrency control*). Existuje i možnost mít *master* server v pohotovostním

stavu (*standby*), kdy na server nepřichází žádné uživatelské dotazy a funguje jen synchronizace s aktivním *master* serverem. *Master* server v pohotovostním stavu se aktivuje poté, co není k dispozici původní *master* server. Tato technologie bývá nazývána logika hlídacího psa (*watchdog logic*).

Databázové replikace se mohou dále rozdělovat podle jejich přístupu k synchronizaci změny mezi servery na synchronní a asynchronní [150, 134]. Synchronní replikace synchronizuje každou změnu okamžitě, a až po přijetí potvrzovací zprávy od *slave* serveru *master* server posílá další změnu. Opak tvoří asynchronní replikace, jež zpravidla synchronizuje balík změn v určitém časovém kroku (každou sekundu, jednou denně, jednou měsíčně) či podle vytížení serverů. Dále lze replikace rozdělit podle toho, zda posílají změny ve formě bitových změn v souboru (fyzická replikace) či posílají daný SQL dotaz a nechají *slave* (*master*) server provést stejnou úlohu.

Téměř každá společnost nabízí vlastní řešení replikačních mechanismů, vlastní pojmenování a funkčnost. Například Oracle nabízí *multimaster* replikaci a replikaci materializovaného pohledu (databázový objekt obsahující výsledek dotazu, včetně dat) [150]. Microsoft SQL Server nabízí snímkovou (*snapshot*), slučovací (*merge*) a transakční (*transaction*) replikaci [67]. MySQL replikace jsou popsány v kapitole 4.1.2.2 na straně 22 a PostgreSQL replikace v kapitole 4.1.2.1 na straně 21. Mezi další technologie pro zvýšení výkonu a robustnosti databázového systému patří rozdělování výkonu (*load balancing*) a techniky vysoké dostupnosti (*high availability*). Tyto techniky spolu s databázovými replikacemi jsou velmi doporučené při budování vysoce robustní a celosvětové služby poskytující prostorová data [108].

Obrázek 3 ilustruje možné přístupy k přenosu dat. První část je centralizovaný databázový server, který poskytuje data všem uzlům. Druhá část ilustruje datovou distribuci napříč všemi uzly v síti.



Obrázek 3: Možnosti přístupů k datům. Zdroj: [108]

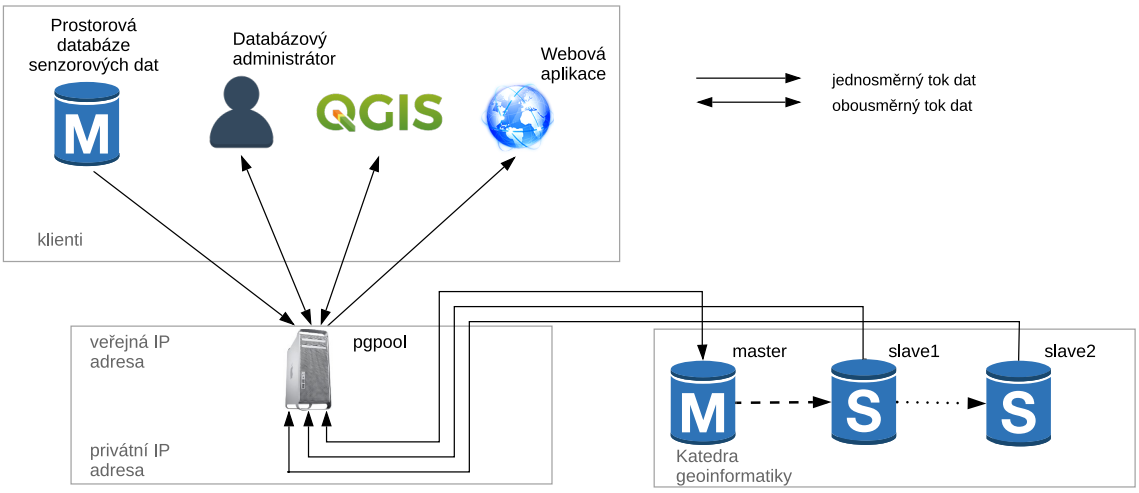
4.2.1 Implementace synchronizačních algoritmů

Tato kapitola se zabývá popisem synchronizačních algoritmů u vybraných databázových systémů. Kritéria pro výběr databázových systémů jsou v tabulce 1 na straně 7. Byla popsána i Esri geodatabáze díky přímé podpoře replikací prostorových dat.

4.2.1.1 PostgreSQL

PostgreSQL od verze 9.0 podporuje synchronní streaming replikaci (fyzická) a posléze i asynchronní replikaci. Od verze 10 nabízí i logickou replikaci, kterou dosud obstarávaly nadstavby pro PostgreSQL. Mezi nejvyužívanější patří PgPool, Bucardo, Postgres-XL a Slony. Slony nadstavba je nejstarší rozšíření pro replikaci PostgreSQL. Fungovala již od verze 7.3 a je dále podporována i ve verzi 12. Slony využívá při replikaci triggery, které monitorují stav sledované tabulky. Slony využívá logickou replikaci (vložení, změna, smazání) a neumožňuje replikaci změny ve schématu. Velkou výhodou Slony je, že lze vytvořit replikaci mezi různými verzemi PostgreSQL na rozdíl od streaming replikace, která vyžaduje totožné *master* a *slave* servery.

PgPool je další nadstavba pro PostgreSQL. PgPool by se mohl charakterizovat jako samostatný middleware pro operační systém Linux, který umožňuje rozložení zátěže, vysokou dostupnost, replikaci, sdílení propojení. PgPool middleware se tváří jako jediný PostgreSQL databázový server, i když sám může obsluhovat více databázových serverů (obrázek 4).



Obrázek 4: Schéma konceptu replikačního řešení.

4.2.1.2 MySQL

MySQL implementuje databázové asynchronní replikace již v základní open-source verzi. MySQL pracuje s „binary log“ souborem, kam se zapisují změny provedené v databázi. Události se do souboru zapisují v různých formátech, podle typu požadované události.

Na tyto události pak reaguje replikační mechanismus, který může být na základě provedeného dotazu neboli SBR (*statement-based replication*), nebo na základě změny v řádku v tabulce RBR (*row-based replication*). Třetí možností je použití kombinace obou metod (*mixed-based replication*). Při replikaci na základě dotazu *master* server zapíše dotaz do *binary log*, ten se synchronizuje do *slave* serveru a následně se dotaz provede i v rámci *slave* serveru. Při režimu RBR zapíše *master* databáze informaci o změnách jednotlivých řádků do „binary log“. Samotná replikace následně kopíruje změny na *slave* server. Režim RBR je nastaven jako výchozí typ replikace pro MySQL. U metody kombinující oba způsoby replikace se automaticky přepíná mezi SRB (zvolená primárně) a RBR v jednotlivých případech. Mimo asynchronní replikace je k dispozici i polosynchronní (*semisynchronous*) a zpožděná (*delayed*) replikace. Polosynchronní replikace zlepšuje datovou integritu, jelikož *master* server čeká na potvrzení transakce od *slave* serveru. Zpožděná replikace určuje *slave* serveru o kolik jednotek času později má provést synchronizaci [14].

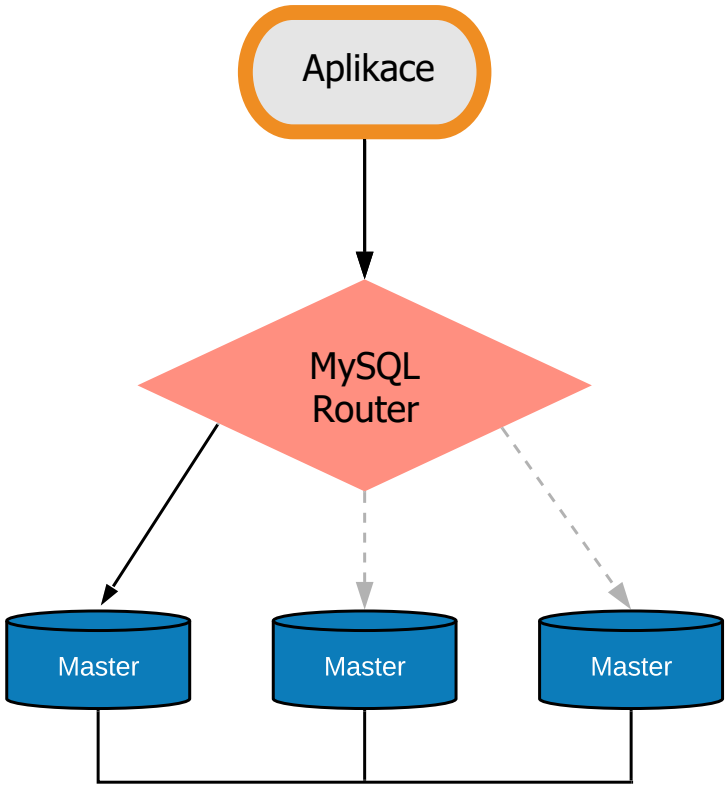
Od osmé verze, využívá MySQL jako defaultní víceúčelové úložiště (*storage engine*) InnoDB. Úložiště poskytuje vyváženost mezi výkonem a spolehlivostí a disponuje všemi, dnes již běžnými, funkcemi, včetně transakčního modelu ACID či využití cizích klíčů pro zachování integrity. Mezi další vlastnosti InnoDB úložiště řadíme například možnost vytvářet distribuované klastry, full-textové vyhledávání a využívá synchronizační metodu MVCC, která je více rozebrána v kapitole 4.2 na straně 30. MySQL umožňuje využívat i další typy úložišť například:

- MyISAM, která bývala výchozí u předchozích verzí MySQL,
- Memory – dříve Heap, uchovává data jen v paměti a hodí se pouze pro dočasná data pouze ke čtení,
- CSV,
- Archive – uchovává velmi úsporně neindexovaná data,
- Blackhole – neukládá žádná data, a tuto tabulku lze využít například pro filtrování dat pro SBR,
- Merge – vytváří datové oddíly,
- Federated – data jsou uložena na vzdáleném serveru a lokálně se nic neukládá,
- NDB – nadstavbové in-memory úložiště speciálně pro vytváření distribuovaných klastrů,
- vlastní.

Pro vytváření distribuovaného prostředí lze využít v MySQL i replikace na úrovni uložišť InnoDB a NDB (*Network DataBase*), které jsou navrženy speciálně pro fungování v klastru. Tyto typy klastrů (InnoDB a NDB klastry) umožňují využití specifických vlastností úložišť. Například NDB rozšiřuje maximální velikost databáze z 64 TB na 128 TB, využívá synchronní replikace v klastru a asynchronní replikace mezi klastry, nebo detekci a opravu chybných uzlů. Oba klastry ovšem využívají funkce replikačních technik z MySQL.

Další využitelnou technologií je MySQL Router. Jeho alternativa pro PostgreSQL je PgPool. Jedná se o placený middleware v rámci MySQL Enterprise, který se chová jako přístupový bod do databáze, a poskytuje spojení mezi aplikací a databázovými servery. Její účel je při výpadku databáze odkázat dotaz na jiný zdroj (*failover*) a pro vyrovnání

zátěže (*load balancing*). Příklad využití je zobrazen na obrázku 5, kde MySQL Router rozděluje dotazy mezi tři *master* servery v *multimaster* replikaci. Uživatel nemusí tušit, k jaké konkrétní databázi se připojuje, to zajistí MySQL Router (či PostgreSQL PgPool). Jedná se o tzv. distribuční transparentnost [126].



Obrázek 5: Příklad použití MySQL Router. Zdroj: [14]

4.2.1.3 Esri geodatabáze

Esri poprvé vydal vlastní řešení geodatabázových replikací pro ArcGIS verzi 9.2, která navazovala na funkcionalitu vzdálené editace z verze 8.3. Pro ArcGIS Pro jsou dostupné replikace od verze 2.5 (vydání Q1 2020). Geodatabázové replikace je možné vytvářet nad osobní i souborovou geodatabází, ovšem vždy ve spojení s ArcSDE. Esri definuje 3 typy replikací:

1. obousměrnou (*Two-way*),

2. jednosměrnou (*One-way*),

3. *checkout / check-in.*

Geodatabázová replikace je postavena na verzovacím systému, který zabezpečuje ArcSDE [122]. Základ pro vytváření geodatabázových replikací je existující podniková geodatabáze (PostgreSQL, Oracle, SQL Server ...) vytvořená pomocí ArcSDE (součástí ArcGIS Enterprise). *Master* geodatabáze, u Esri označována jako rodič (*parent*), poté musí mít nastavenou funkci globálního identifikačního čísla. To z důvodu, aby každý prvek v geodatabázi vlastnil jedinečný identifikátor. Data musí být uložena ve vysoké přesnosti, i když tato podmínka platí jen pro geodatabáze starší než verze 9.2. Novější geodatabáze pracují pouze ve vysoké přesnosti, tedy prostorová vrstva pokrývající celý svět může mít body v blízkosti až 10 nanometrů (nízká přesnost umožňovala přiblížit body asi na 2 centimetry). Poslední podmínkou k vytvoření geodatabázových replikací je povolení verzování geodatabáze.

Slave databáze, u Esri potomek (*child*), závisí na typu replikace. U obousměrné geodatabázové replikace musí být *master* i *slave* geodatabáze řízena ArcSDE. U checkout / check-in replikace stačí mít *slave* geodatabázi jako souborovou geodatabázi. U jednosměrné geodatabázové replikace je rozhodující směr synchronizace změn. *Master* geodatabáze musí být vždy podniková geodatabáze pod ArcSDE. Pokud je směr synchronizace z *master* geodatabáze do *slave* geodatabáze, může být *slave* geodatabáze souborová geodatabáze. Pokud synchronizace směřuje ze *slave* do *master* geodatabáze, tak *slave* geodatabáze musí být podniková geodatabáze využívající ArcSDE middleware.

Databázová replikace a geodatabázová replikace se nesmí zaměňovat. Esri využívá verzovacího mechanismu middleware ArcSDE a nezávisí na možnostech backend databáze (např. PostgreSQL). Replikace následně umožňuje práci i s komplexními datovými typy, jako jsou geometrické sítě či topologie.

Velkou výhodou využití Esri geodatabázových replikací je schopnost pracovat s komplexními datovými typy a řešení prostorových konfliktů. Nevýhodou je nutnost využívat ArcSDE, respektive ArcGIS Enterprise. Data také nejsou synchronizována okamžitě a automaticky, ale až po vyvolání akce ze strany uživatele (klikem na tlačítko).

4.2.2 Řešení konfliktů při synchronizaci

Při synchronizacích mohou vznikat konflikty. Například, pokud aplikaci stačí práce s kopií dat pouze pro čtení, žádný konflikt při synchronizaci vzniknout nemusí. To je ovšem ideální případ, aplikace často potřebuje do databáze zapisovat. Poté je nasnadě neotevírat celou tabulku i pro zápis, ale například pouze některé sloupce.

Klasické konflikty při synchronizaci jsou [150]:

- konflikt aktualizací,
- konflikt odstranění,
- konflikt unikátních hodnot.

Konflikt aktualizací vzniká při současné editaci (*update*) záznamu. Může nastat v případě, kdy se dvě transakce z různých zdrojů snaží editovat stejný řádek ve stejnou dobu. Konflikt odstranění může nastat, pokud první transakce maže záznam v první databázi a druhá transakce edituje či maže záznam ve druhé databázi. Druhá transakce nahlásí chybu, jelikož záznam již neexistuje. Konflikt unikátních hodnot nastane tehdy, pokud dojde ke vložení záznamů se stejným primárním klíčem do synchronizovaných databází. Tyto konflikty se vyskytují pouze při *multimaster* replikaci (o *multimaster* replikaci více zde [134, 14, 150]).

Konflikty se mohou vyskytovat v centralizovaném databázovém i distribuovaném prostředí [126], kdy každá architektura má svá specifika. Základní dva koncepční algoritmy pro řešení, respektive předcházení kolizím pomocí vynucené izolace pro centrali-

zované databázové prostředí jsou, zamykání (*locking*) a časová razítka (*timestamping*). Při kontrole souběžnosti dvou či více transakcí může být využito zamykání k izolování záznamu, které využívá jedna transakce. K zamykání může docházet na úrovni záznamu, řádku, tabulky, databáze. Mezi obecné přístupy řešení kontroly souběžnosti pomocí zamykání řadíme jednofázové a dvoufázové zamykání a fantomův jev [126]. Zamykání v RDBMS nejčastěji využívá principy:

- zámek konverze (*Lock Conversion*),
- zámek povýšením (*Lock Upgrade*),
- zámek degradováním (*Lock Downgrade*),
- zamykání vícečetné granularity (*Multiple-Granularity Locking*),
- zámek průběhu (*Intention Locking*).

Časová razítka se využívají k seřazení transakcí a jejich operací v plánovači databázových úloh (vnitřní systém DBMS). Obecně se tomuto přístupu říká řazení časových razítek (*timestamp ordering*). Časové razítko nemusí být pouze skutečný čas, ale může být využit logický čas, který představuje posloupnost celých čísel. Jednotlivé algoritmy kontroly souběhu pomocí časových razítek jsou základní (nebo také agresivní), konzervativní, multiverzová. Právě multiverzová kontrola (*MVCC*) je jednou z nejpoužívanějších. Systém ukládá každou hodnotu i všechny její historické změny jako její verze. Kdykoliv se transakcí mění data, vytvoří systém novou verzi těchto dat. Pokud tedy nastala změněna dat n-krát, pak mají data n verzí. Nemusí se tedy blokovat žádné transakce pro čtení, vždy se čte nejnovější verze dat.

Algoritmy pro kontrolu souběžnosti transakcí lze klasifikovat i na základě množství konfliktů, které se v databázovém prostředí mohou vyskytnout. Pokud systém vykazuje velkou míru konfliktů, využije se pesimistický algoritmus pro kontrolu souběžnosti (*pessimistic concurrency control*). S nízkou mírou konfliktů se využije optimistická kontrola

(*optimistic concurrency control*). Při pesimistické kontrole souběžnosti, tedy v systémech s vysokou mírou konfliktů je snaha identifikovat a synchronizovat transakce co nejdříve, aby se minimalizovalo možné opakování celé transakce. U optimistického algoritmu je to naopak, tedy nastává snaha pozdržet synchronizaci transakce a opět zabránit opakování transakce, která často vyžaduje i další zdroje serveru.

V prostředí distribuovaných databázových systémů se používají podobné principy jako pro centralizované databázové systémy [126]. Musí se ovšem rozšířit o globální kontrolu, která určí pořadí transakcí v celém distribuovaném prostředí. Kontrola může probíhat centralizovaně na daném databázovém uzlu. Globální transakce ovšem musí pracovat pouze s tímto uzlem. Případně musí probíhat distribuovaná kontrola, pokud transakce probíhá na více uzlech současně. Mezi používané kontrolní mechanismy patří:

- centrální dvoufázové zamykání (*Centralized two-phase locking*),
- primární kopie dvoufázového zamykání (*Primary Copy two-phase locking*),
- distribuované dvoufázové zamykání (*Distributed two-phase locking*),
- distribuovaná kontrola součinnosti časových razítek (*Distributed Timestamp Concurrency Control*).

V navrhované architektuře proudových dat výše zmíněné konflikty nehrozí. Senzorová data přichází z různých zdrojů do Python middleware, kde jsou okamžitě přeposlána do jednotné tabulky. U Zigbee a Meshlium brány proudí data přes externí MariaDB databázi a následně jsou data replikována do jednotné prostorové databáze. Veškerá data jsou vkládána do jednotné tabulky, která je až následně replikována v distribuovaném prostředí. Takto nemohou vzniknout konflikty, jež by se musely automaticky či ručně řešit a opravovat. Data ze senzorů se zpětně neopravují. Vše řeší Python middleware, který detekuje a vyhodnocuje nepřesná měření a do jednotné tabulky tato data vůbec nezaznamená.

4.3 Senzorové sítě a IoT

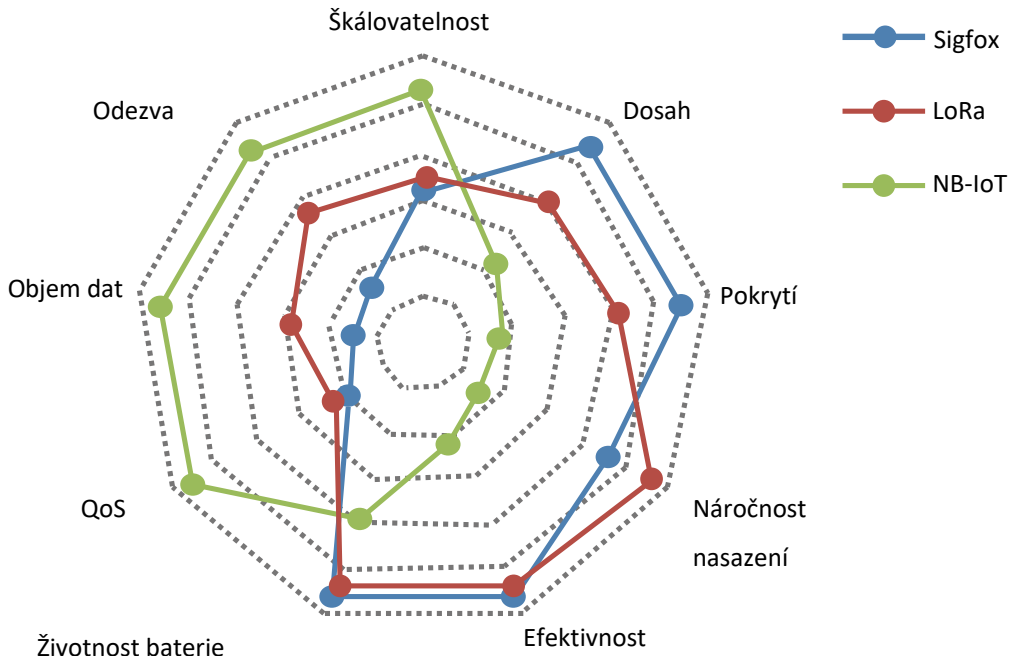
Senzory a senzorové sítě jsou producenty automatizovaných strojových dat. Každým dnem roste jejich počet, a tím roste i objem vyprodukovaných dat. Pro analýzu těchto dat je potřeba jejich uložení. Pro tento účel jsou téměř výhradně využívány databáze. Díky heterogenitě senzorů a senzorových sítí, ale také různorodosti způsobu, jak data získávají, odesílají, a jakým způsobem jsou nakonec tato data analyzována a prezentována, je potřeba navrhnout robustní distribuovanou prostorovou databázovou síť, robustní middleware a aplikační server pro integraci dat. Následně poskytnout uživatelům nejen data, ale nabídnout jim také možnost z těchto dat získat potřebné informace [70].

Senzorová síť je infrastruktura, která zahrnuje pozorování či měření, zpracování a komunikaci, jež dává administrátorovi možnosti řízení, pozorování a reagování na události a jevy ve specifickém prostředí [140]. Základní čtyři předpoklady senzorové sítě jsou a) kolekce lokalizovaných senzorů; b) propojená síť (ne nutně bezdrátově); c) centrální sběrný bod (komunikační brána); d) výpočetní zdroje brány i dalších prvků k ověřování správnosti dat, k dotazování, řízení událostí a data mining [140]. Senzorovým sítím se velmi detailně věnovala Mgr. Vendula Hejlová, Ph.D. ve své disertační práci [69].

Základem a původní technologií, ze které se postupně část konceptu IoT vyvinula, je M2M (*Machine to Machine*) komunikace. M2M komunikace začínala v 60. letech 20. století, kdy byla poprvé použita pro ID volajícího. Jednotné definování M2M nebo IoT je velmi složité a existuje velké množství různých definic a pohledů [72]. Mezinárodní telekomunikační unie [6] definuje IoT jako globální infrastrukturu pro informační společnost zpřístupňující pokročilé služby propojením (fyzických i virtuálních) věcí na základě existujících i vyvíjených interoperabilních informačních a komunikačních technologií. Na tvorbě konceptu IoT se podílelo nespočet existujících aplikací s významným dopadem na každodenní aktivity [63]. Například: inteligentní přepravní systémy, obchodní/procesní řízení, e-zdraví [43]. IoT může sloužit také pro úsporu zdrojů či předcházení poruch

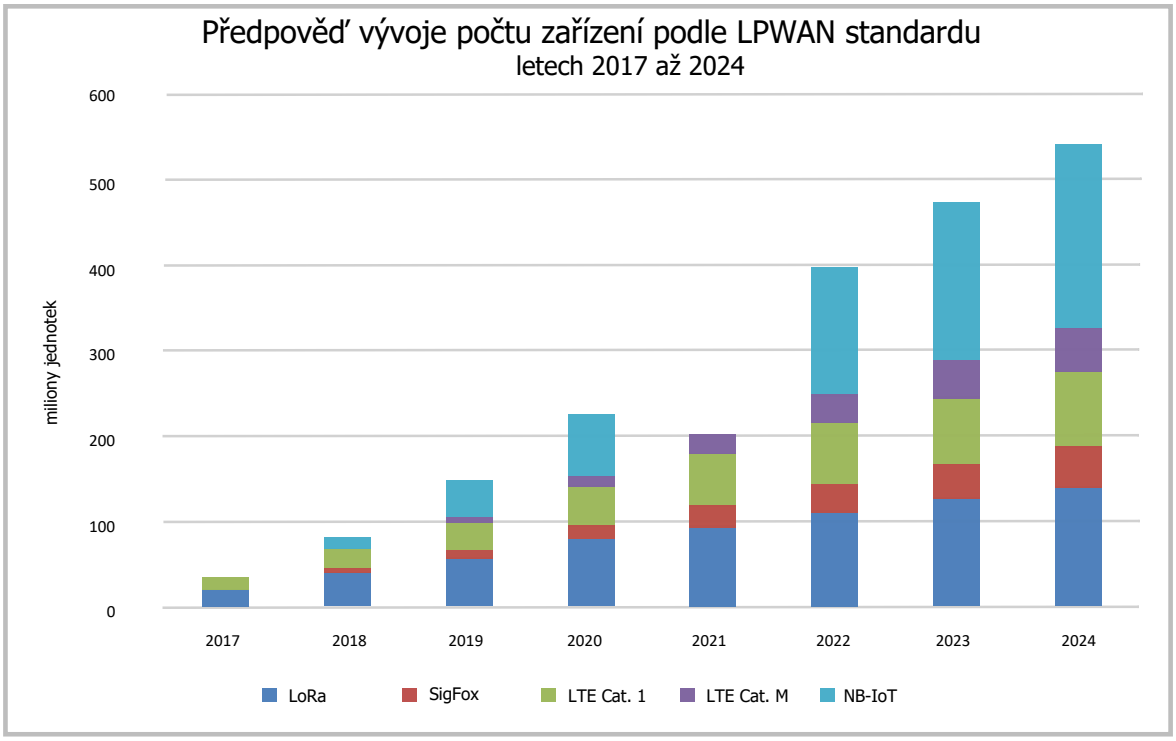
(*predictive maitanance*). Je to další technologická vlna, která od základu mění fungování podnikatelských procesů a praktik napříč téměř všemi průmyslovými a sociálními sektory. Jedná se například o energetiku, průmyslovou výrobu, přepravu zboží či zdravotní péči. To, co IoT dokáže změnit, je, že přidává další rozměr světa strojů a věcí k již existujícímu Internetu a to přidáním senzorů k zachycení fyzických vlastností a akčních prvků k jejich ovládání. Myšlenka konceptu IoT je o možnosti zpřístupnění inteligentní činnosti zahrnující stroje a věci reálného světa, ať již IoT pracuje pro koncového zákazníka, firmu či výrobní průmysl. Inteligentní činnost využívají software k získávání dat o reálném světě. Ze získaných dat lze získat nové informace a znalosti k její následné automatizaci [148].

S rozšiřujícím se počtem aplikací, dochází ke standardizaci IoT oblastí, především pak v komunikaci a získávání dat. Se vzrůstajícím objemem standardů a protokolů vznikají další nové architektury. IoT využívá nejen starších protokolů, např. 6LoWPAN, IPv4(6) či komunikačních technologií, např. Wi-Fi, Bluetooth, Zigbee, GSM, ale i speciálně zaměřených na IoT např. LoRa, Sigfox, IQRF, NB-IoT, MQTT, DDS (*Data Distribution Service*), CoAP (*Constrained Application Protocol*) [41, 72, 80, 119, 124, 149].



Obrázek 6: Porovnání technologií Sigfox, LoRa a NB-IoT. Zdroj: [102]

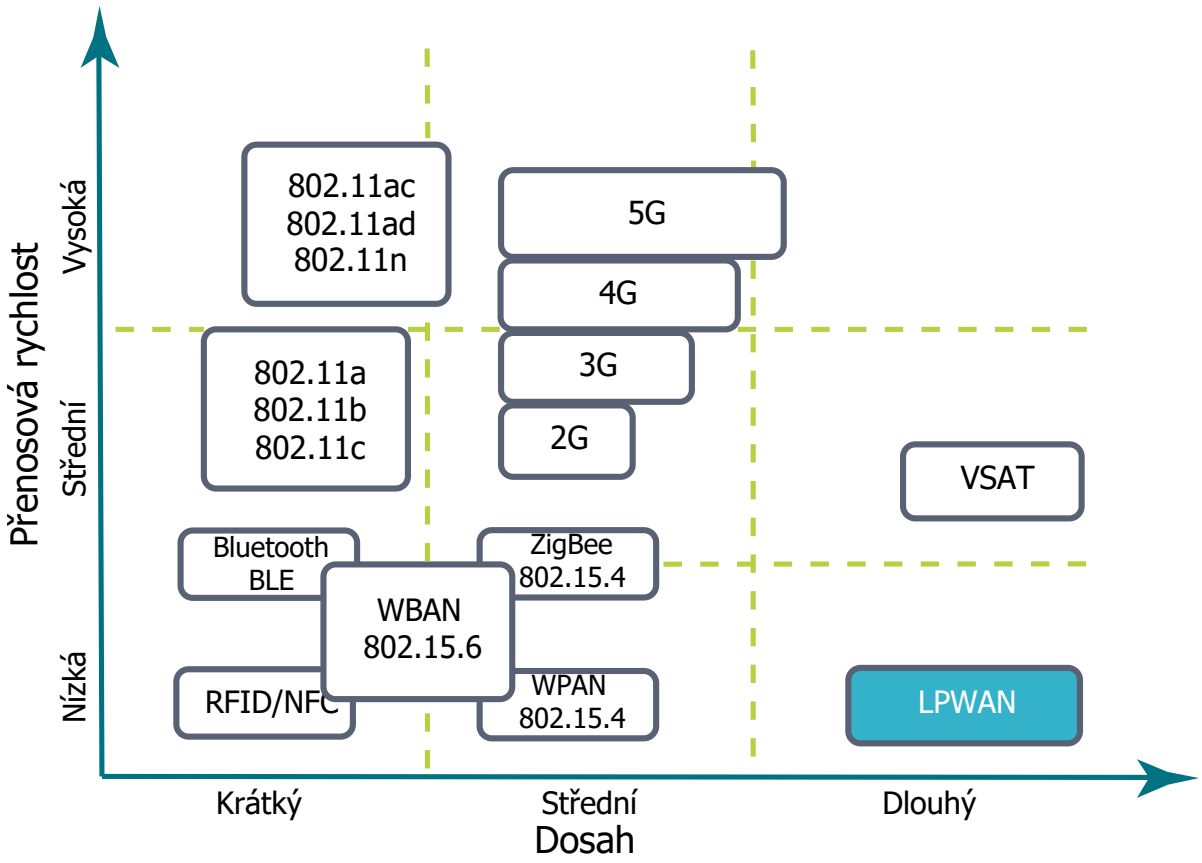
IoT komunikační technologie z oblasti LPWAN negarantuje QoS (*Quality of Service*) a nejsou vhodné pro hard real-time aplikace [132]. NB-IoT je z pohledu zajištění QoS vhodnější nežli Sigfox a LoRa, jejichž doménou je spíše dlouhá výdrž, nízká pořizovací cena a flexibilní rozsah projektů (od jednotek koncových zařízení ve vlastní LoRa síti až po statisíce, například odběrná místa vody). Význam IoT si uvědomují i u společností McKinsey a Gartner, které předpokládají obrovské využití těchto technologií v reálném životě [32, 62, 98]. Předpovídají, že v roce 2025 bude mít IoT ekonomický vliv mezi 3,9 – 11,1 bilionů amerických dolarů [148]. Pro představu, IoT má potenciál až 11 % světové ekonomiky. Je tedy předpoklad, že bude zapojováno stále více senzorů a geoprostorové analýzy budou mít větší význam než dnes (obrázek 7).



Obrázek 7: Předpověď vývoje počtu zařízení LPWAN mezi lety 2017 a 2024. Zdroj: [109]

4.3.1 Komunikační protokoly pro přenos senzorových dat

Rozmachu Internetu věcí, pro jehož nasazení je často důležitá spotřeba energie, resp. dlouhodobá (v měsících) výdrž na baterii, bylo potřeba upravit i komunikační technologie. Jednou z nejstarších globálních komunikačních technologií je GSM (*Groupe Spécial Mobile*).



Obrázek 8: Diagram komunikačních technologií. Zdroj: [16]

V dnešní době je GSM spolehlivá technologie s dobrým pokrytím území, ovšem velmi energeticky náročná, a tedy nevhodná pro IoT zařízení napájené pomocí baterie. Z tohoto důvodu vznikají v posledních letech nové komunikační technologie, které se mohou zařadit pod společnou skupinu LPWAN (*Low-Power Wide-Area Network*) nebo LPN (*Low-Power Network*). V České republice jsou momentálně tři celorepublikové LPWAN sítě a to technologie Sigfox poskytovatele SimpleCell, LoRaWAN poskytované Českými Radiotelekomunikacemi a technologie NB-IoT (*Narrow Band – Internet of Things*) poskytovaná mobilními operátory O2, T-Mobile a Vodafone. Mimo celorepublikových či celosvětových (*NB-IoT*) existují i lokální sítě od rozlohy pole až po pokrytí měst. Vlastní LPWAN síť je možné vybudovat pomocí technologií LoRa nebo IQRF.

IQRF platforma vznikla v České republice a je určena pro bezdrátovou komunikaci s nízkým výkonem, nízkou rychlostí a nízkým objemem dat. IQRF využívá síťovou topologii mesh, kdy se data z nejvzdálenějšího uzlu (mezi uzly může být vzdálenost jednotky až stovky metrů) postupně přeposílají přes jednotlivé uzly až k bráně (maximálně 240 skoků).

IoT kromě velmi nízkých energetických nároků potřebuje také upravené komunikační protokoly, jelikož se spotřebovanou energií souvisí i délka odesílané zprávy. Čím je zpráva delší, tím musí být i delší vysílací čas, a tím narůstá spotřeba. S délkou zprávy také klesá šance, že zpráva dorazí bezpečně k přijímající bráně. Např. Sigfox má maximální délku zprávy 12 bytů, NB-IoT až 1kB, IQRF až 64 bytů a u LoRaWAN není velikost definována a rychlost odesílání záleží na vzdálenosti a kvalitě signálu mezi uzlem a bránou, ale běžně se pohybuje v rozmezí od 300 bps do 50 kbps [116, 4]. LPWAN tedy zaujímají místo tam, kde při nízkém výkonu dokáží operovat na kilometrové vzdálenosti a přitom je přenosová rychlost v rozsahu desítek bitů za sekundu až jednotky kilo-bitů za sekundu [31].

Mimo LPWAN se mohou využívat i PAN, tedy technologie krátkého dosahu mezi které patří například NFC (*Near field communication*), Bluetooth, Zigbee či Wi-Fi. Technologie GSM a Wi-Fi patří do kategorie energeticky velmi náročných technologií. Senzory s touto komunikační technologií jsou vhodné zejména tam, kde je senzor připojen ideálně k permanentnímu zdroji energie. Senzory komunikující pomocí GSM může samotný GSM modul v proudové špičce odebírat až 2 A, což je 100× až 1000× více, než v případě LPWAN technologií.

Komunikační technologie se mohou dělit i podle dostupnosti frekvenčních pásem na licencovaná a nelicencovaná frekvenční pásma. GSM a NB-IoT využívají licencovaná pásma. V nelicencovaném pásmu pracují Sigfox, LoRa, LoRaWAN, IQRF, které využívají pro Evropu frekvenční pásmo 868 MHz, pro USA 916 MHz a v rámci celého

světa frekvenci 433 MHz. V těchto pásmech může kdokoliv vysílat, ovšem jen po omezenou dobu. Zpravidla může uživatel využívat 1 % času, kdy „zatěžuje“ frekvenční pásmo (864 s za den). Při průměrné délce odesílání jedné zprávy 6 s (Sigfox, LoRa, LoRaWAN) je možné odeslat maximálně 140 zpráv za den. To je 1 680 B za den (pro Sigfox), tedy 1,64 kB dat za den.

4.3.1.1 Komunikační architektury

V roce 2015 vydal IAB (*Internet Architecture Board*) dokument zabývající se návrhem Internetem propojených chytrých zařízeních [147]. Diskutují celkem čtyři různé přístupy toho, jakým způsobem může zařízení komunikovat. Základní komunikační vzory jsou: Zařízení – Zařízení, Zařízení – Cloud, Zařízení – Brána a Cloud – Cloud.

Komunikace Zařízení – Zařízení je přímá komunikace mezi dvěma či více zařízeními. Může se jednat o zařízení různých výrobců, které pak ovšem musí podporovat konkrétní standardy, například na které fyzické vrstvě budou komunikovat (LoRa, Bluetooth, IEEE 802.15.4 (*Institute of Electrical and Electronics Engineers*)) či jaký použijí aplikační protokol (CoAP, MQTT, DDS). Zde je nejdůležitější dodržování standardů.

Komunikace Zařízení – Cloud pak využívá toho, že je zařízení schopno přímo posílat data na aplikační server. Zařízení je připojeno pomocí Wi-Fi, Ethernetu či mobilního operátora přímo k Internetu.

Komunikace Zařízení – Brána je pak nejčastějším architektonickým vzorem pro budování IoT LPWAN sítí. Brána představuje most mezi technologiemi a komunikačními platformami. Brána je zařízení s permanentním napájením a přístupem k Internetu. Umožňuje přijímat data ze senzorů pomocí určité technologie (např. LoRa, Sigfox, Zigbee) a přeposílat data dále na aplikační servery pomocí další technologie (Ethernet, Wi-Fi, GSM). Brána může být specializovaný hardware, který funguje například pod RTOS (*Real-time operating system*), nebo zcela univerzální zařízení typu Raspberry Pi.

Posledním typem komunikačních vzorů je Cloud − Cloud, který rozšiřuje vzor Zařízení − Cloud. Pro některé aplikace je nutné využít data z více zdrojů a umožnit tak přístup k datům i dalším aplikačním serverům. Typicky je přístup vytvářen pomocí RESTfull API a dovoluje tak třetí straně získat data.

4.3.1.2 LoRa, LoRaWAN

LoRa je technologie fyzické vrstvy sloužící k vytvoření komunikačního spojení na velkou vzdálenost [94]. LoRaWAN je komunikační protokol a síťová architektura celé sítě. Zařazení LoRa a LoRaWAN v rámci referenčního modelu ISO/OSI je uvedeno na obrázku 9. LoRa technologii vyvinula společnost Cycleo, kterou v roce 2012 zakoupila firma Semtech a je jediným výrobcem čipů s LoRa modulací. LoRaWAN protokol je otevřený a spravuje jej LoRa Alliance.

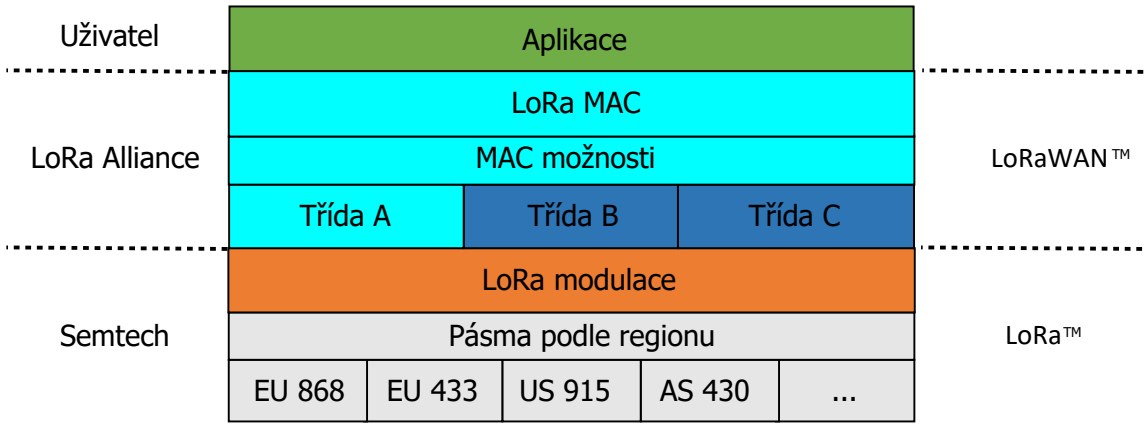
LoRa moduluje data rozprostřeným spektrem respektive jeho modifikací Chirp Spred Spectrum, která byla vyvinuta pro vojenskou a vesmírnou komunikaci. LoRaWAN využívá topologii hvězdy-hvězd (*Star of stars*), kde jsou uzly propojeny přes jednu čí více bran. Brány následně přeposílají zprávy na síťový server, který by měl být „nejchytřejší" z celé architektury LoRaWAN [29].

Síťový server by měl být schopen monitorovat brány a koncová zařízení, agregovat příchozí zprávy, přeposílat zprávy na konkrétní aplikační servery a řešit duplicitní zprávy (základ je v RSS – *Recieved Signal Strength*). V případě LoRaWAN v České republice se o síťový server starají České Radiotelekomunikace a přeposílají data ve formátu JSON pomocí POST na SSL zabezpečený server. CRa budují LoRaWAN celorepublikovou síť od roku 2016.

Oproti uzlům, které napájí především baterie a 99 % času jsou v režimu spánku, jsou brány napájeny nepřetržitě a jsou připojeny k vysokorychlostnímu Internetu. Existují i soukromé brány, které nabízí podobnou funkcionalitu. Není zde ovšem garance, že brána bude vždy k dispozici, a že data proudící přes bránu jsou zabezpečena.

Nejznámějšími platformami jsou The Things Network a LoRIOT. Jednou z největších výhod LoRa je, že si uživatel může vybudovat vlastní síť. LoRaWAN brána může být vytvořena i z běžného Raspberry Pi s nadstavbou pro LoRaWAN. Pomocí LoRa technologie mohou spolu jednotlivé uzly komunikovat a tím vytvořit mesh topologii.

Další technologií založenou na využití vlastních bran je IQRF a Zigbee. U těchto technologií by bylo velmi obtížné pokrýt signálem větší území z důvodu jejich krátké komunikační vzdálenosti.

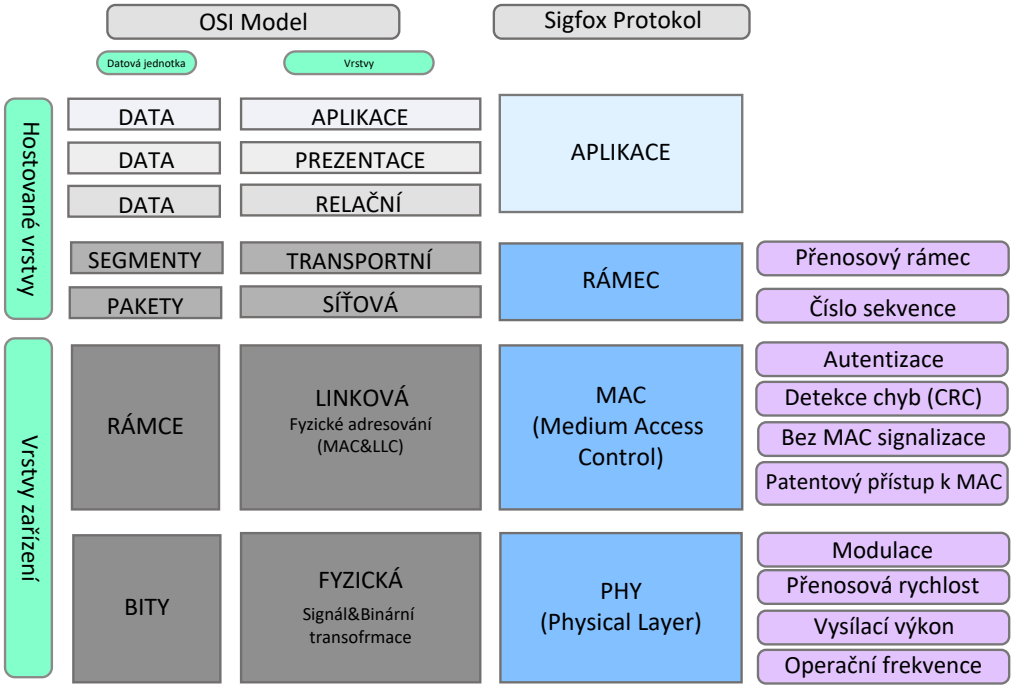


Obrázek 9: Architektura LoRa podle OSI modelu. Zdroj: [95]

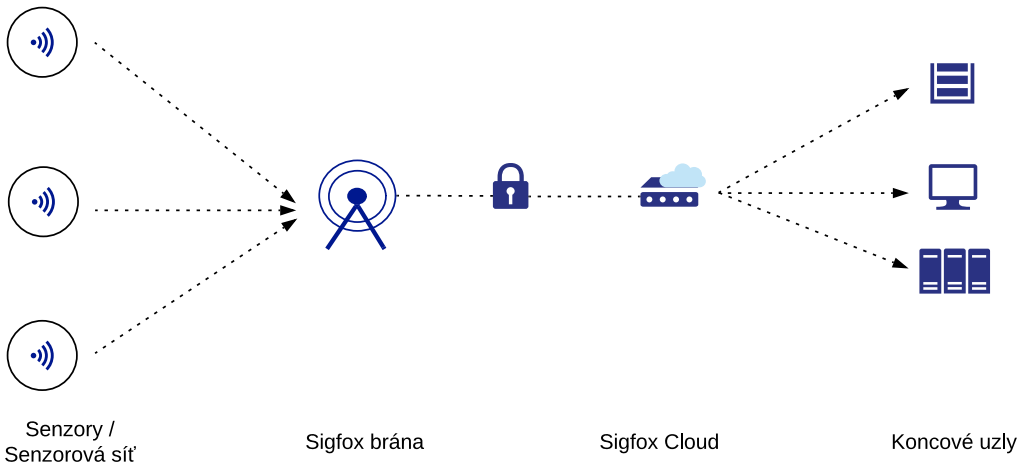
4.3.1.3 Sigfox

Sigfox je francouzská firma, která vyvinula stejnojmennou technologii pro IoT. Technologie Sigfox je postavena na komunikační technologii ultra úzkého pásma (*Ultra Narrowband*). Firma Sigfox licencuje hardware a poskytuje licenci na výstavbu sítí operátorům. V České republice provozuje síť Sigfox operátor SimpleCell. Ten se spojil s firmou T-Mobile, která jí umožnila vybudování vlastních Sigfox základnových stanic (*BTS – Base Transciever Station*) na již existujících základnových stanicích T-Mobile. Dnes již tato síť pokrývá 96 % České republiky [17]. Sigfox umožňuje odeslat 12 B uživatelských dat, jedná se tedy například o kombinaci 12 celých čísel v hexadecimálním formátu, tedy v rozsahu 0 – 255 v decimálním formátu, nebo 12 písmen v ASCII (*Ame-*

rican Standard Code for Information Interchange) kódu. Pro zvýšení pravděpodobnosti, že zpráva dorazí na bránu v pořádku se zpráva odesílá 3× ve třech různých náhodných frekvencích v definovaném spektru. Na obrázku 10 je srovnání vrstev Sigfox s referenčním modelem OSI. Na obrázku 11 je znázorněna obecná architektura Sigfox komunikace.



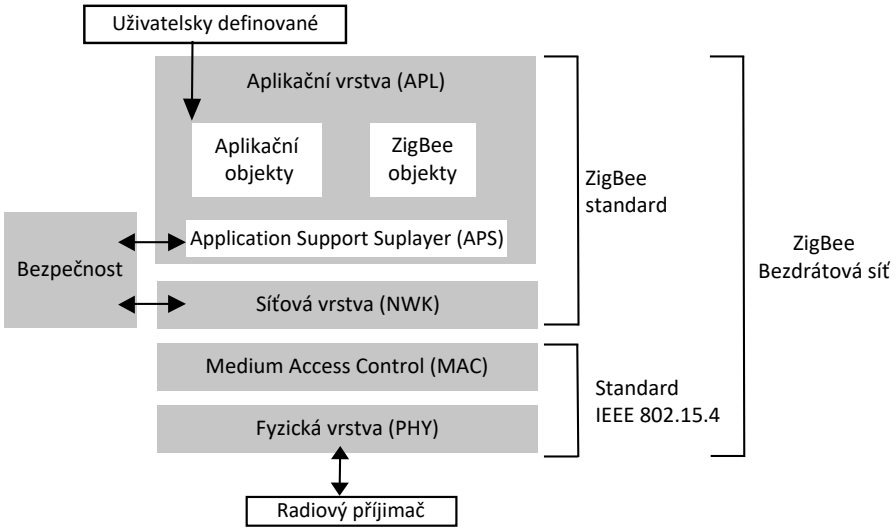
Obrázek 10: Vrstvy technologie Sigfox ve srovnání s OSI modelem. Zdroj: [53]



Obrázek 11: Architektura Sigfox

4.3.1.4 Zigbee

Zigbee technologie byla jednou z prvních, která se zaměřila na nízko objemové a nízko energetické přenosy. Její základ tvoří technologie IEEE 802.15.4. Může komunikovat v nelicencovaných pásmech 868 a 433 MHz, i v pásmu 2,4 GHz. Technologie ovšem umožňuje komunikaci senzor – brána na poměrně malou vzdálenost (100 – 300 m). Proto se řadí spíše mezi PAN spolu s Bluetooth a RFID. Základ IEEE 802.15.4. umožňuje komunikaci v topologiích hvězda a P2P (*Peer-to-Peer*) [151]. Zigbee technologie využívá ke komunikaci s koncovými body bránu. V červnu 2019 vyšla nová specifikace Zigbee 3, označovaná jako Zigbee Pro 2017 (R22). Ta vychází ze Zigbee Pro 2015 (R21) a je zpětně i dopředně kompatibilní [20]. Hlavní doménou Zigbee budou převážně vnitřní prostory, kde brána dokáže pomocí 2,4 GHz pokrýt velký prostor i přes překážky. Na obrázku 12 jsou znázorněny vrstvy Zigbee protokolu podle referenčního systému OSI. Senzory komunikující na technologii Zigbee se detailně věnovala Mgr. Vendula Hejlová, Ph.D., ve své disertační práci, kde popisuje možnosti a principy komunikace senzorů Libelium na technologii Zigbee.



Obrázek 12: Vrstvy protokolu Zigbee pro bezdrátovou síť. Zdroj: [151]

4.3.1.5 NB-IoT

Narrowband IoT má základ v již existující LTE (*Long Term Evolution*) technologii. NB-IoT vyvíjí 3GPP skupina, která v rámci 13. vydání zavedla podporu pro velmi jednoduchá zařízení s velmi malou datovou propustností [8]. Nasazení NB-IoT proběhne pouze v aktualizaci firmware základnových stanic (*BTS*). Z původního LTE bylo odstraněno několik funkcionalit, které jsou pro IoT zbytečné. Jedná se například o monitorování sítě, agregaci či vzájemné obousměrné propojení. NB-IoT využívá stejné licencované frekvence jako LTE a využívá QPSK (*Quadrature phase-shift keying*) a BPSK (*Binary-Phase Shift Keying*) digitální modulaci. Na rozdíl od LoRaWAN a Sigfox se NB-IoT připojuje pouze k jedné bráně. Pokud ztratí bránu, ke které se pravidelně připojuje, například kvůli pohybu v prostoru, snaží se vyhledat novou bránu [29]. Další rozdíl mezi NB-IoT a LoRaWAN je, že NB-IoT má téměř čtyřnásobný špičkový odběr proudu při odesílaní a pětinásobný odběr ve spánku. NB-IoT má výhody ve vyšším QoS, nižších latencích a delší komunikační vzdálenosti, ovšem za vyšší cenu i spotřebu [139].

4.3.1.6 IQRF

IQRF je další LPWAN, respektive LPPAN (*Low Power Personal Area Network*), protože z výsledků reálného testování technologie nedosahuje takových přenosových vzdáleností [61]. Využívá nelicencovaná pásma shodná pro LPWAN. Komunikuje na vzdálenost několika desítek metrů a posílá velmi malé objemy dat (64 B) rychlostí 20 kb za sekundu. Největší síla IQRF je v mesh síti, kde si uzly přeposílají zprávu až k bráně (koordinátor). Brána dokáže obsluhovat až 239 uzlů. Každý paket (2 b − 64 B) dokáže skočit až 240× a zvládne 33 skoků v jedné sekundě [4]. Mesh topologie IQRF vznikla v roce 2008 a byla nasazena v mnoha projektech [33, 117, 156]. Pokud nemůže uzel komunikovat s bránou, lze vytvořit i stromovou strukturu, ve které nejvzdálenější uzel posílá data přes bližší uzly směrem k bráně. Takto lze vytvořit metropolitní senzorovou síť ve stovkách m^2. Nevýhodou mesh sítě je časová prodleva (jednotky s).

4.3.2 Přenos dat v reálném čase

Podle prof. Buttazza [44] je hlavní rozdíl mezi real-time a non real-time systémem ten, že real-time systém musí úspěšně dokončit úkol v daném časovém intervalu. V real-time systému je výsledek po daném termínu nejen opožděný, ale i nebezpečný. V závislosti na důsledcích z nedodržení daného maximálního termínu (času zpracování) lze systémy ještě dále rozdělit na tvrdé (*hard*), měkké (*soft*) a přísné/blízké (*firm/near*). U hard real-time systémů může mít minutí daného maximálního času katastrofické následky. Jedná se například o protiblokovací systém (*Anti-lock Brake System − ABS*) či obchodování na burze. Pokud soft real-time systém nedodrží stanovený čas, způsobí degradaci výkonu, ale neohrozí to stabilitu celého systému. Firm/near real-time systém je systém mezi hard a soft real-time. Systém může několikrát minout daný maximální čas zpracování a nezpůsobí to pád či selhání systému. Po častějším zpracování požadavku po daném čase se systém může zhroutit. Podle [103] jsou tyto definice pro systémy zabezpečující bezpečnost, ovšem pro Big Data analýzy jsou tyto definice poněkud uvolněny a real-time je spíše schopnost zpracování dat, jakmile přijdou do systému. Někdy jsou real-time analýzy pro Big Data označovány jako near real-time, tedy v čase blízkém reálnému času.

Real-time označení pro systémy a procesy se zaměřuje kromě správnosti výsledku a správného vyhodnocení pro akční člen také na čas dokončení úkolu. Proudová data (*Stream data*) jsou speciální v tom, že data proudí neustále od většího množství zdrojů a nejčastěji v malých objemech (například v rámci B až kB) [21]. Pokud systém bude zpracovávat data například dvě minuty, ale nová data budou proudit každou minutu, systém za poměrně krátkou dobu přestane být real-time, a nakonec se dostane do fáze, kdy se budou muset některá data zahodit, aby mohl opět plnit svou funkci [55].

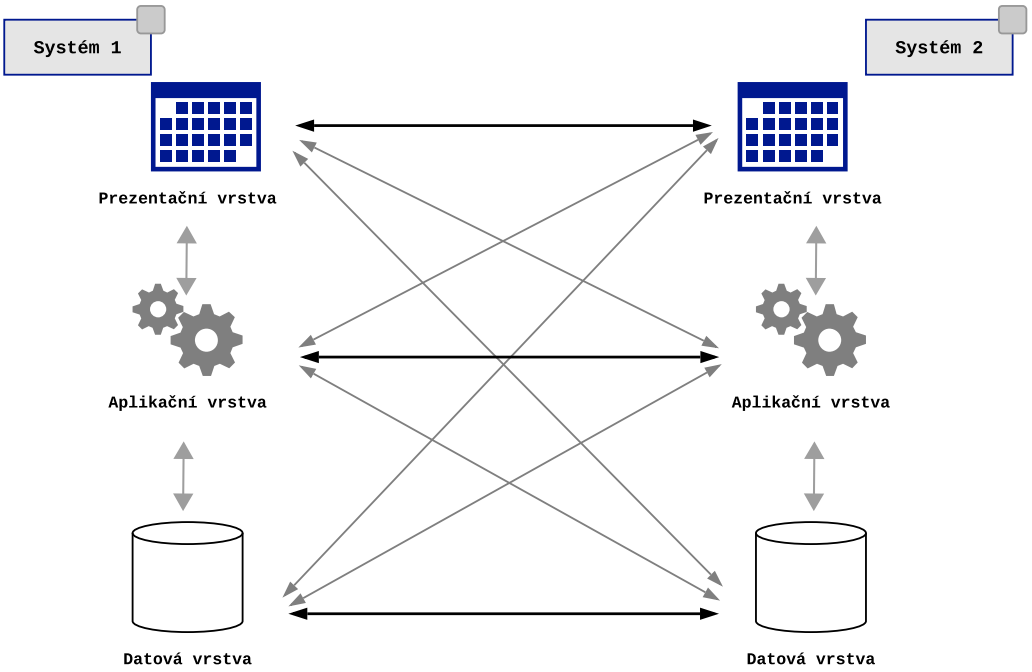
Základní vlastnosti proudových dat je, že proudí neustále z různých zdrojů (např. různé senzorové sítě měřící teplotu), mají volnou a měnící se strukturu a nakonec se databáze musí vypořádat s vysokou kardinalitou dat, tedy s vysokou mírou různých hodnot, které mohou záznamy obsahovat. Na proudová data můžeme nahlížet jako na real-time nebo non real-time data. Toto dělení závisí na účelu využití těchto dat.

Dnes jsou tato data běžně zpracovávána pomocí dávkového zpracování. Data jsou uložena v databázi, a operátor vezme část či celý dataset a provádí nad ním analýzy. To lze jednoduše provést nad daty, která mají podobu historických dat, která byla sesbírána v určitém časovém horizontu. Práce s proudovými daty je odlišná a je zapotřebí nepřerušovaná analýza nad stále nově přicházejícími daty [111]. Proudová data se mohou zpracovávat záznam po záznamu či pomocí posuvného časového okna [21].

4.3.3 Integrace senzorových dat

Obecný pojem „integrace“ můžeme chápat jako propojování počítačových systémů, společností nebo lidí [71, 106]. Počet aplikací a systémů se ve výrobní společnosti, ale i v běžné praxi neustále zvyšuje a každá firma si chrání své know-how. Integrace mezi těmito systémy bývá obtížná a zvyšuje finanční i pracovní náročnost na správu a samotnou práci s vyšším množství systémů [143]. Z pohledu architektury informačních systémů, lze integraci aplikovat v různých vrstvách (úrovních):

- uživatelské rozhraní – vznikají portálová řešení;
- aplikační vrstvy – sjednocení aplikačních logik do tzv. těsné vazby;
- datové vrstvy – centralizací či propojením datových zdrojů různých systémů;
- mezi různými vrstvami – vytváření služeb, které následně komunikují mezi vrstvami.



Obrázek 13: Integrace na různých vrstvách architektury informačních systémů. Zdroj: [83]

Integrace se může rozdělit do šesti obecných integračních přístupů, které navrhují jednotlivé způsoby komunikace mezi jednotlivými systémy. Jsou jimi:

- informační portály – spojení různých systémů do jediné webové aplikace;
- datové replikace – synchronizace společných datových skladů;
- sdílené funkce – využívání jedné funkce více aplikacemi;
- architektura orientovaná na služby (*Service-Oriented Architectures*) – princip společných sdílených služeb, které jsou vyhledatelné a mají jasný popis vstupně-výstupních parametrů;
- distribuované procesy – jedna transakce je rozšířena mezi různé systémy;
- podniková integrace (*Bussiness-to-Business Integration*) – komunikace mezi různými podniky, např. dodavatelem a příjemcem.

Každý z těchto přístupů řeší odlišnou oblast, kterou je potřeba propojit se stávající či nově budovanou aplikací tak, aby zapadala do celého konceptu podniku, strategie a aplikace. Tyto přístupy nemají ostře vymezenou hranici použití a často se jejich definice částečně překrývají. Integrace bývá často velmi problematická. Nejjednodušší možností je přímá úprava stávající aplikace tak, aby se mohla začlenit do stávajícího celku. Bohužel, většinou z důvodů licenční politiky nebo finanční náročnosti jsou úpravy aplikace neproveditelné. Navíc zde vzniká riziko nekompatibility dalších závislých podsystémů. Proto vzniká koncept EAI (*Enterprice Application Integration*), který využívá k integraci heterogenních systémů „prostředníka“ (*middleware*). Middleware zabezpečuje příjem, transformaci a směrování zpráv a také správu a monitorování monolitických aplikací [118].

Samotnou aplikační integraci lze vyřešit pomocí čtyř integračních stylů [71]:

- přenosem souborů – aplikace vytváří soubor v definovaném formátu, který si navzájem posílají;
- sdílenou databází – aplikace ukládají data do sdílené databáze;
- vzdáleným voláním procedur – aplikace nabízí služby (většinou synchronní komunikace), které mohou využít ostatní aplikace;
- zprávami – aplikace provádí výměnu dat či ovlivňují své chování pomocí výměny zpráv (asynchronní komunikace).

V heterogenní senzorové síti poté lze využít procesu, kdy jeden typ senzorové sítě posílá data v určitém jednotném datovém formátu. Síť poté lze považovat za homogenní aplikaci, která komunikuje pomocí výměny zpráv. Následná integrace je provedena pomocí middleware, jež umožní zpracování zprávy senzorové sítě tak, že jsou uloženy do jednotné (prostorové) databáze. Tato databáze je následně replikována a tím je zajištěna integrace dat bez její centralizace.

Nejběžnější způsoby aplikační integrace lze řešit pomocí základních topologií. Jedná se především o integraci *Point to Point*, *Broker Based*, *Publisher − Subscriber*, *Message Bus* [125]. Mezi nejjednodušší se řadí *Point to Point*, kde je každý systém přímo spojen s každým systémem. Tím roste exponenciálně počet nutných rozhraní, ale i finanční a administrativní náročnost. *Broker Based* nebo *Middleware Based* používá softwarového prostředníka který přeposílá jednotlivé zprávy od odesílatele přímo k příjemci. Middleware může současně ověřit validaci zprávy, autorizaci a autentizaci odesílatele, a může provést i transformaci zprávy tak, aby ji příjemce mohl bezpečně přijmout. V některých případech by mohl middleware znamenat úzké hrdlo celého systému. Dnes již jsou middleware vytvářeny tak, aby bylo možné jejich horizontální škálování. *Publisher − Subscriber* integrace není postavena na základě adresy příjemce, ale na základě obsahu zprávy. K příjemci se pak dostávají zprávy pouze z témat, ke kterým jsou přihlášeni k odběru. K příjemci se dostává obsah, jakmile je publikován zdrojem. *Message Bus* je principiálně podobný *Broker Based* a *Publisher − Subscriber* integraci. Z *Broker Based* využívá myšlenku prostředníka pro zpracování zpráv a z modelu *Publisher − Subscriber* pak využívá tzv. adaptéry, které se starají o příjem zpráv a komunikují přímo s konkrétním systémem. Veškeré tyto základní topologie aplikační integrace pracují tak, že jsou zachovány téměř beze změny aktuální systémy. Je spíše ekonomickou otázkou, zda se vyplatí využít výše zmíněné topologie či úplně změnit architekturu integrace, například pomocí ESB (*Enterprise Service Bus*) v rámci SOA (*Service Oriented Architecture*, více o SOA v kapitole 4.4.3 na straně 68). Hlavní ovlivňující faktory jsou: a) počet systémů; b) míra integrace jednotlivých systémů; c) počet budoucích systémů k integraci; d) náklady na údržbu; a za e) náklady na vybudování nové architektury integrace [125].

Pro integraci senzorových dat se nejčastěji používá middleware nebo *Publisher − –Subscriber* koncept. Senzory komunikující pomocí LPWAN technologií se vždy snaží odesílat co nejmenší objem dat. Nelze proto počítat s žádnými složitějšími strukturami

dat, např. které by byly samopopisné (XML) či obsahovaly metadata. Národní LPWAN sítě (Sigfox, LoRaWAN) navíc nabízí možnost získání dat jen pomocí REST API či nejrůznějších push based technologií (*callback*) v JSON formátu. Následně je tedy nutné data přijmout a zpracovat. Jako middleware může sloužit klasický webový server (více v kapitole 5.3 na straně 100) napsaný například v jazyce Python, Java, PHP, nebo lze využít některý z open-source či komerčních IoT platforem.

Pro LPWAN existuje mnoho IoT platforem z oblasti PaaS (*Platform as a Service*), SaaS (*Software as a Service*). Některé IoT platformy jsou nabízeny také k instalaci na vlastní server (*on-premise*). Mezi nejběžnější PaaS IoT platformy patří Azure Event Hub, Azure IoT Hub, Amazon Kinessis Data Streams, Amazon IoT, IBM Watson IoT, Google Cloud IoT Core. Mezi SaaS například Oracle IoT, Thingsboard, ThingSpeak, Kaa, Macchina, Ubidost, SensorUP (první implementace OGC SensorThings API z celkových čtyř certifikovaných) či Loriot. Tento výčet IoT platforem není konečný a každým dnem se rozrůstá. Platformy nabízí multiplatformní, multiprotokolový, často distribuovaný, škálovatelný a modulární systém. Tyto platformy ovšem nenabízí žádnou běžnou funkcionalitu z oblasti GIS. Často zobrazují data v grafech, tabulkách, ukazatelích, a pokud umí přijímat i prostorové souřadnice, dokáží je pouze zobrazit nad mapou (například sledování autobusu). Firma Esri vydává GeoEvent Server, který umí přijímat zprávy a nabízí základní možnosti zpracování (oblast (*buffer*), výpočet pole (*field calculator*) a 23 dalších) dat s přímým napojením na ArcGIS Online. Licencuje se ovšem nad rámec běžného ArcGIS Server.

Integrace senzorových dat byla zvolena pro řešení disertační práce na datové vrstvě a za pomoci databázových replikací. Senzorová data mohou být často integrována až pomocí aplikační vrstvy či vrstvy uživatelského prostředí. Tyto integrace ovšem nejsou vhodné pro další zpracování v prostředí GIS. V GIS softwarech je často nutné uchovávat data v jednotném formátu a mnohdy i ve specifické struktuře, aby software mohl s daty

pracovat. Senzorová data z různých komunikačních technologií (Sigfox, Zigbee, GPRS) byla přes middleware posílána do jednotné databáze. Aby data nebyla centralizována na jednom místě, je databáze distribuována pomocí databázových replikací. Celá distribuovaná databázová síť je uzpůsobena prostorovým datům s cílem poskytovat prostorová data do desktop GIS i pro web (více v kapitole 5.3 na straně 100).

4.4 Koncept Sémantického Webu

Sémantický Web představil jako první Sir Tim Berners-Lee v roce 2001 [39]. Sám autor řekl: „Snažit se používat Sémantický Web bez SPARQL je jako snažit se používat relační databáze bez SQL“. Jedná se o další evoluci tzv. Webu Dokumentů. W3C, starající se o standardizaci také mluví o Sémantickém Webu jako o Webu Dat [153]. Idea Sémantického Webu je poskytovat sady technologií a standardů, jež umožňují strojové porozumění webovému dokumentu (stránce) [159]. Následně pak automaticky provádět úkoly, které by musely být splněny manuálně a na velkém množství webů. Sémantický Web může být chápán jako významový web.

Technologie a protokoly, které utvoří Sémanticky Web, jsou RDF (*Resource Desciption Framework*), GRDDL (*Gleaning Resource Descriptions from Dialects of Languages*), RDFa (RDF pro XHTML), OWL2 (*Web Ontology Language*), JSON-LD (*JSON-based Serialization for Linked Data*), SKOS (*Simple Knowledge Organization System*), RDFS (*RDF Vocabulary Description Language 1.0: RDF Schema*), POWDER (*Protocol for Web Description Resources*), PROV (*interchange of provenance information on the Web*), RIF (*Rule Interchange Format*), SAWSDL (*Semantic Annotations for WSDL and XML Schema*), RDB2DF (*Relational Databases to RDF*), SHACL (*SHapes And Constraints Language*) [154]. Další koncept přímo utvářející Sémantický Web je Linked Data (Propojená Data). Ta jsou publikována a propojena pomocí technologií a standardů Sémantického Webu.

4.4.1 SPARQL

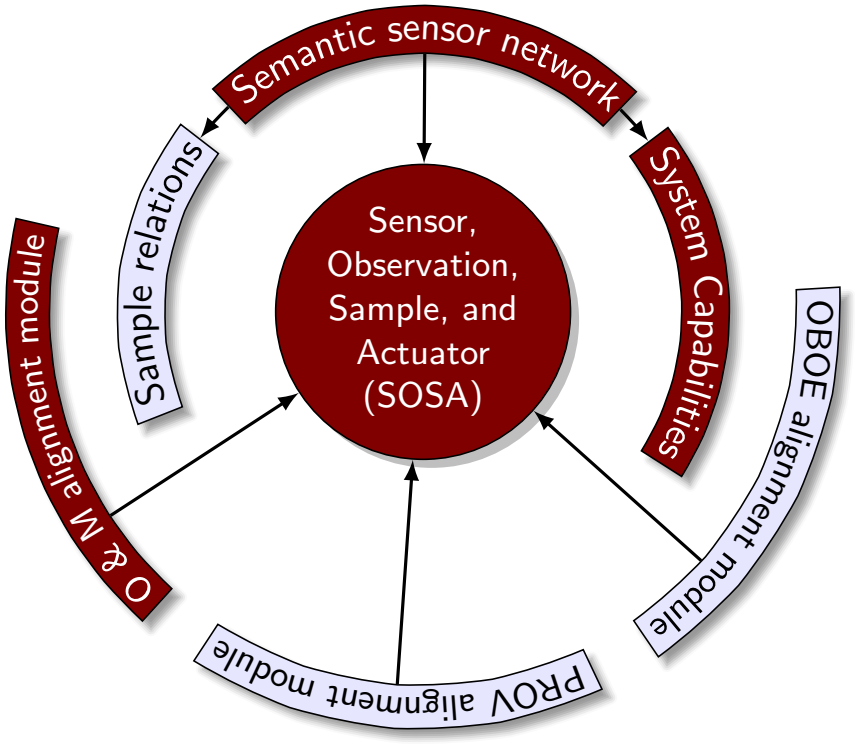
SPARQL je rekurzivní zkratka pro SPARQL Protocol and RDF Query Language [115]. Jedná se o sadu specifikací, která umožňuje dotazování a manipulování s RDF grafovým obsahem na webu či RDF úložišti. Pro propojení uzlů v RDF úložišti (trojce subjekt, predikát, objekt) se má výhradně používat HTTP IRI (URI v UTF-8) a tím jednoznačně identifikovat zdroj [101]. Specifikaci vydalo W3C SPARQL Working Group poprvé 15. 1. 2008 s verzí SPARQL 1.0. Standard verze 1.1 byl vydán 21. 3. 2013. SPARQL 1.0 oproti verzi 1.1 neměl funkci INSERT do RDF struktury. Standard 1.1 obsahuje celkem 11 dokumentů. Původní 3 dokumenty, které tvořily standard, byly SPARQL Query Language specification, jež tvoří jádro SPARQL. SPARQL Query XML Result Format specification, který popisuje XML formát pro serializaci výsledku SPARQL dotazu. Nakonec SPARQL Protocol for RDF specification, jež využívá WSDL 2.0 pro využití HTTP a SOAP protokolu pro vzdálené dotazy do RDF databází. SPARQL je jednou z komponent pro Sémantický Web [159]. Na Webu či v samostatné aplikaci vznikají tzv. SPARQL endpoints, terminály, které zpřístupňují RDF datasety lidem a strojům. SPARQL endpoint může být nakonfigurován tak, aby vracel výsledky v různých datových formátech, například v HTML či XML.

OGC GeoSPARQL je standard rozšiřující stávající SPARQL sadou funkcí, RIF pravidel a RDF/OWL slovníkem pro geografické informace se základem v General Feature Model, Simple Feature, Feature Geometry a SQL MM [115]. GeoSPARQL rozšiřuje SPARQL atributové dotazy na dotazy s geometrickou složkou [115]. První verze byla vydána 27. 10. 2009 a poslední 19. 7. 2012. GeoSPARQL 1.0 byl vytvořen pro dotazování nad prostorovými daty v prostředí Sémantického Webu. GeoSPARQL definuje slovník pro reprezentaci prostorových dat v RDF a rozšiřuje SPARQL dotazovací jazyk pro práci s prostorovými daty [101]. GeoSPARQL obsahuje komponenty pro práci s prostorovými daty, kde základ tvoří RDFS/OWL třídy pro prostorová data a dále topologický slovník

definující RDF možné vlastnosti a dotazy na topologické vztahy mezi prostorovými objekty. Geometrická komponenta definuje RDFS datové typy pro serializaci geometrických dat, RDF vlastnosti spojené s geometrií a netopologické prostorové dotazy a dále komponenta pro dotazování na geometrickou topologii. GeoSPARQL bohužel neumožnuje práci s rastry a je tedy odkázán jen na vektorová data ve formátu WKT. Framework Ontop s extenzí Ontop-Spatial umožňuje připojení relačních prostorových databází jako například PostgreSQL, SpatiaLite či Oracle. Mimo jiné umožňuje i práci s rastrovými daty a umí komunikovat s nadstavbou Rasdaman (PostgreSQL), MonetDB či SciDB [36]. Ontop pracuje se standardem R2RML, který připojuje OWL ontologii k relační databázi a pomocí R2RML Mapping Engine vytváří RDF úložiště. Práci s rastry upravil W3C až 28. 11. 2017, kdy představil standard Data Cube. I tak jsou ale rastry uloženy v jiném formátu než Data Cube (primárně zaměřen na multidimenzionální statistická data) či RDF, a je tedy nutné vytvářet virtuální grafy pomocí middleware [42].

Na švédské univerzitě ve městě Lund byla testována výkonost pěti nejběžnějších RDF úložišť s podporou GeoSPARQL (RDF4J, GeoSPARQL-Jena, Virtuoso, Stardog a GraphDB) [75]. Během testování vyvstala řada problémů např. nejsou nepodporovány některé prostorové dotazy či je podporován pouze WGS84 (*https://epsg.io/4326*) souřadnicový systém (jen GeoSPARQL-Jena podporuje souřadnicové transformace). Největší problém ovšem tvořil nejednotný výsledek na stejný dotaz, kdy referenční Esri ArcGIS 10.3.1 vrátil 10 výsledků na dotaz, GeoSPARQL na dotaz vrátil 11 výsledků. Z testovaných databázích jen GraphDB vrátilo správný výsledek. Hlavní příčina může být v rozdílné strategii pro práci s přesností v jednotlivých databázích. Nicméně tato nekonzistence je problém a i z této studie vyplývá že RDF úložiště musí projít dalším vývojem pro širší nasazení pro GIS.

Společný projekt W3C-OGC Spatial Data on the Web Working Group (*SDWWG*) standardizovalo 19. 10. 2017 Ontologie Sémantického Senzorového Webu (*Semantic Sensor Network Ontology – SSN*). Ten popisuje senzory, sledované jevy, objekty a jejich vlastnosti, zapojené procedury a akční členy [68]. Základem ontologie je SOSA (*Sensor, Observation, Sample, Actuator*), která obsahuje základní třídy a vlastnosti. Obě ontologie dokáží popsat rozsáhlou skupinu aplikací a objektů, jako jsou satelitní snímky, velkoplošné monitorování, průmyslové a domácí infrastruktury, nebo Web Věcí. Starší standard OGC SWE (*OGC Sensor Web Enablement*) poskytl prostředky k popisu senzorů a jejich měření. Nicméně tento standard není možné použít s technologiemi W3C Sémantického Webu [77]. Proto byly představeny ontologie SSN a SOSA, které jsou postaveny tak, aby poskytovaly flexibilní a zároveň koherentní perspektivu pro reprezentaci entit, vztahů a aktivit spojených se snímáním, vzorkováním a akcí (obrázek 14).



Obrázek 14: SOSA a její vertikální a horizontální moduly

Na obrázku 14 lze vidět modulární rozložení v horizontální i vertikální segmentaci. Vertikální segmentace přidává další hloubku axiomatizaci přímým importem nižších modulů a definování nových axiomů. Horizontální segmenty rozšiřují ontologickou oblast například novými třídami a relacemi pro specifický podporovaný systém, ale již více nerozšiřují sémantiku existujících termínů. Před finálním vydáním W3C a OGC Recommendation byly diskutovány příliš silné ontologické závazky a celá ontologie byla pro koncept Webu Věcí nadměrně těžkopádná. Celý koncept SSN tedy šel proti trendu velmi štíhlých slovníků, jež preferovaly Propojená Data (*Linked Data*) a největší server s ontologiemi Schema.org. Navržené modulární rozložení dovolilo publikování dat na web i na Web Věcí velmi širokému publiku [77].

W3C Web Věcí (*WoT – Web of Things*) je vytvářen k získání interoperability napříč IoT platformami a aplikačními doménami. WoT nabízí mechanismy k formálnímu popisu IoT rozhraní tak, aby byly služby a zařízení spolu schopny nezávisle komunikovat, ať již na základě jejich základní implementace či síťového protokolu [85]. Tvůrci této architektury byly zaměstnanci firem Huawei, Fujitsu, Oracle, Panasonic a Hitachi. Lze tedy předpokládat, že vytvořená architektura bude v budoucnu využívaná největšími firmami na světě, a bude se velmi rychle rozvíjet a implementovat. WoT architektura navíc nabízí standardizovanou cestu jak definovat a programovat chování IoT. WoT architektura má čtyři základní stavební bloky. Jsou jimi WoT Thing Description, Binding Templates, Scripting API a Security and Privacy Considerations. Dokument WoT Thing Description je označen jako Candidate Recommendation a Scripting API jako Working Drafts. Stále se tedy nejedná o finální standard a dokumenty čeká ještě schvalovací proces Proposed Recommendation a nakonec W3C Recommendation [130]. Paralelně s W3C připravuje Ben Francis z Mozilla Corporation dokument, který popisuje běžný datový model a API pro Web of Things [59]. Ten vedle popisu „Věci“ pomocí klasického JSON také popisuje konkrétní HTTP a WebSockets protokol pro Web of Things, jež pracuje

s již existujícími webovými technologiemi. Nenutí tak webové prohlížeče implementovat nové API, mnoho newebových IoT protokolů či RDF úložiště. K zajištění interoperability „Věcí“ je nutný společný univerzální slovník a běžné API tak, aby nebylo potřeba proprietárních datových formátů a nového API.

Publikování (prostorových) dat v konceptu Sémantického Webu je velmi vhodné pro veřejné datové sady ministerstev a státních podniků tak, aby měli všichni uživatelé možnost získat data jednotným způsobem v jednotném formátu. Jedná se i o univerzální formát pro harmonizaci dat mezi velkými poskytovateli dat. Tam, kde není potřeba veřejného publikování dat s co nejširším dopadem na celou společnost, bude velmi těžké obhájit nasazení takto robustního konceptu. Ovšem, při budování nových veřejných dat či senzorových sítí (*Smart City, Smart Traffic*) by se mělo již na koncept Sémantického Webu pamatovat. Tyto veřejné zdroje by mohly využívat jednotlivci i firmy k vytváření nových aplikací s nejrůznějšími analýzami s využitím Propojených Dat. Sémantický Web je navíc velmi mladý koncept, který čeká na budování ontologií, slovníků a vhodných RDF úložišť pro prostorová data. Sémantický Web lze následně vybudovat nad distribuovanou geodatabází pomocí např frameworku Ontop s extenzí Ontop-Spatial, který vytváří virtuální RDF graf nad relační prostorovou databází (PostgreSQL, SQLite, Oracle) pomocí průběžné GeoSPARQL-to-SQL transformace [38, 37, 100]. Dalším příkladem může být EU projekt GeoKnow [88]. Tříletý projekt mezi roky 2012 a 2015 měl za cíl transformovat geografická data do prostředí sémantického webu [89]. Projekt navazuje na projekt LinkedGeoData [78], který převedl OpenStreetMap do RDF. Nicméně na GitHub již projekt nevykazuje více než čtyři roky žádnou aktivitu.

Udržování dat v RDF databázích a dotazování přes GeoSPARQL není momentálně předními softwarovými producenty GIS software (Esri, QGIS) podporováno. Projekty, které transformovaly prostorové informace uložené v relačních databázích do RDF úložišť, používaly především překladače SPARQL-SQL s ontologií a mapovacím slovníkem.

Tato technika může sloužit jako middleware ke klasickému konceptu ukládání prostorových dat do relačních databází, jež mají majoritní podporu v GIS software. Publikování prostorových geodat v rámci konceptu Sémantického Webu je zaměřeno především na jednoduchou vizualizaci či aplikaci základních prostorových dotazů přes GeoSPARQL. Pokročilé prostorové dotazy a dynamická geovizualizace stále patří do domény relačních a NoSQL dokumentových databází.

4.4.2 OGC SensorThings API

SensorThings API od OGC je otevřený standard, který podporuje přenos prostorových informací a je navržen tak, aby usnadnil naplňování konceptu IoT zavedením společného datového modelu a rozhraní pro komunikaci mezi zařízeními navzájem a zařízeními a aplikacemi přes síť Internet [90]. Pro toto propojení je možné využít architektonický styl REST pomocí RESTful API nebo pomocí modelu komunikace *Publisher − Subscriber* přes protokol MQTT (*Message Queuing Telemetry Transport*) [10]. Data jsou přenášena pomocí datového formátu JSON (algoritmus 1). SensorThings API má základ v OGC SWE a v ISO Open Data Protocol (*OData*). Díky využití datového formátu JSON přenášeného pomocí RESTful webových služeb nebo protokolem MQTT je tento standard vhodný i pro implementaci časoprostorových senzorových dat, a byl prověřen studiemi a projekty [65, 46, 145, 74]. Oproti OGC SWE SOS, která implementuje protokol pro vzdálené volání služeb SOAP a pro přenos dat využívá datový formát XML, které jsou však implementačně náročnější a režie spojená s přenosem je větší, je implementace přenosu dat pomocí RESTful nebo MQTT a datového formátu JSON vhodnější [73].

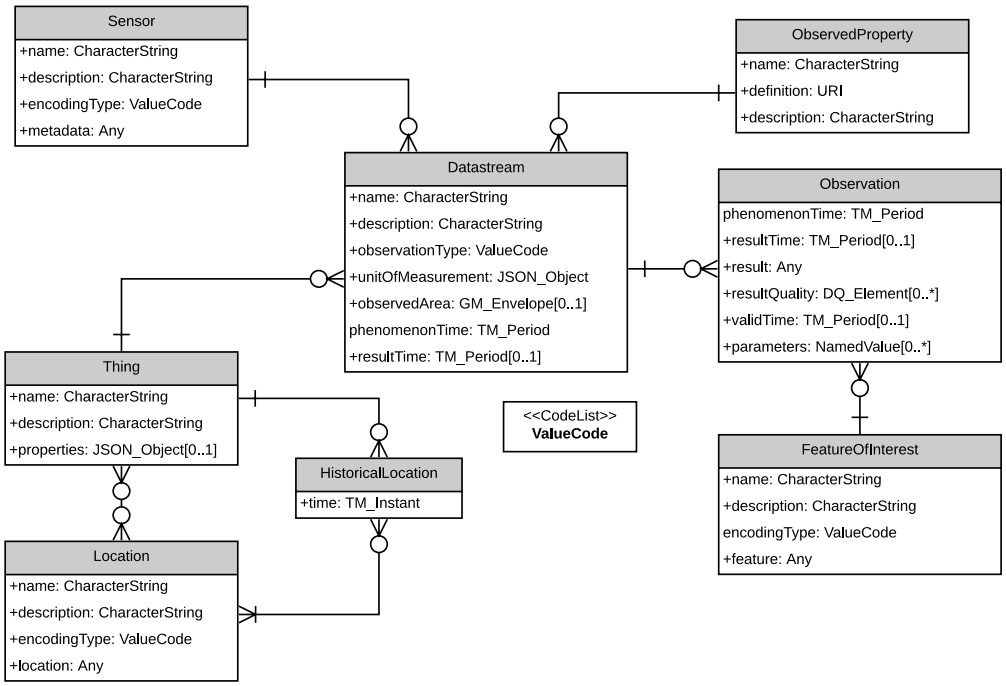
SensorThings API se skládá ze dvou částí. Jedná se o část Snímání (*Sensing*) [90] a část Úkolování (*Tasking*) [91]. Část standardu o snímání definuje způsob zápisu dat pro přenos ze senzorů (obdoba *OGC SOS*), druhá část naopak definuje, jakým způsobem úkolovat senzory a aktivovat akční členy (obdoba *OGC SPS*). Datový model

SensorThings API (obrázek 15) byl navržen tak, aby fungoval spolu se starším datovým modelem OGC Observations and Measurement, který je součástí OGC SWE. Plně certifikované implementace OGC SensorThing API Part 1: Sensing 1.0 mají celkem čtyři organizace. Jsou jimi Fraunhofer Institute of Optronics, System Technologies and Image Exploitation, Karlsruhe, Německo; GOST, Geodan, Nizozemsko; Institute of Communication and Computer Systems (*ICCS*), Atény, Řecko; a nakonec SensorUp, Calgery, Kanada. Německá a nizozemská implementace je zdarma stažitelná na serveru GitHub [60, 64].

Aby mohl OGC SensorThings API standard kooperovat s evropskou směrnicí INSPIRE, musí být provedeny určité změny v datovém modelu. Ty jsou již diskutovány a navržené změny datového modelu by mohly být implementovány v následující verzi SensorThings API 1.1 [84]. Další studie se zabývá způsobem vyřešení automatické registrace senzorů k bráně [73] v rámci OGC SensorThing API protokolu. Datový model musel být v tomto případě rozšířen o další prvky.

Algoritmus 1 Příklad implementace OGC SensorThings API – dotaz na Datastream. Zdroj: [135]

```
{
  "@iot.id": 206051,
  "@iot.selfLink": "https://toronto-bike-snapshot.sensorup.com/v1.0/Datastreams(206051)
      ",
  "description": "The datastream of available docks count for the Toronto bike share
      station Bloor St / Brunswick Ave",
  "name": "7061:Bloor St / Brunswick Ave:available_docks",
  "observationType": "http://www.opengis.net/def/observationType/OGC-OM/2.0/
      OM_CountObservation",
  "unitOfMeasurement": {
    "symbol": "{TOT}",
    "name": "dock count",
    "definition": "http://unitsofmeasure.org/ucum.html#para-50"
  },
  "Observations@iot.navigationLink": "https://toronto-bike-snapshot.sensorup.com/v1.0/
      Datastreams(206051)/Observations",
  "ObservedProperty@iot.navigationLink": "https://toronto-bike-snapshot.sensorup.com/v1
      .0/Datastreams(206051)/ObservedProperty",
  "Sensor@iot.navigationLink": "https://toronto-bike-snapshot.sensorup.com/v1.0/
      Datastreams(206051)/Sensor",
  "Thing@iot.navigationLink": "https://toronto-bike-snapshot.sensorup.com/v1.0/
      Datastreams(206051)/Thing"
}
```



Obrázek 15: Datový model OGC SensorThings API

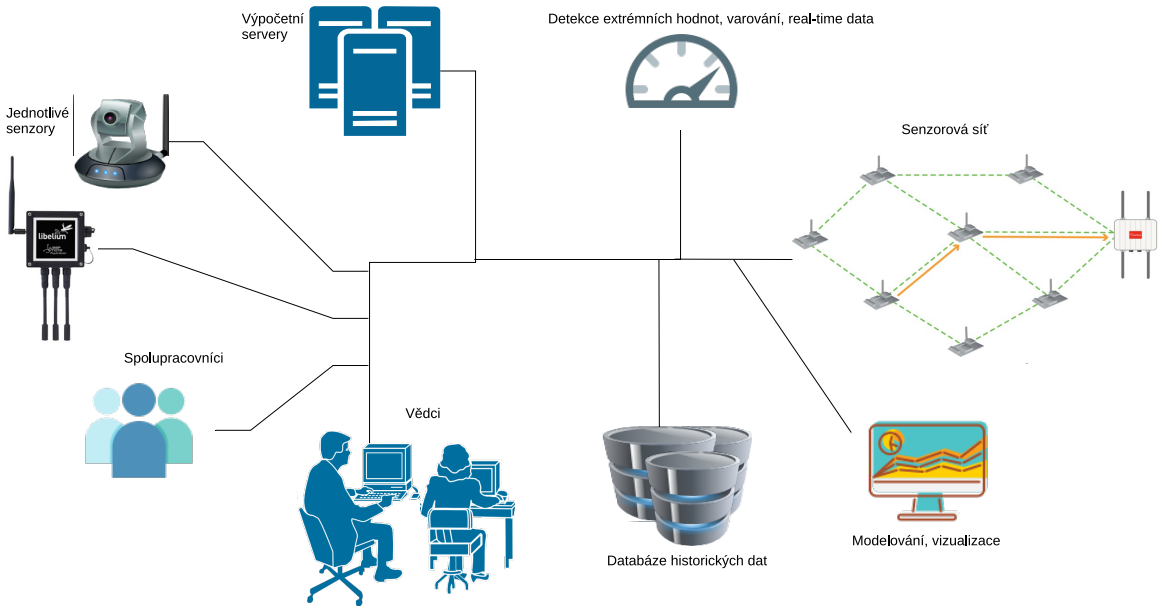
4.4.3 Sensor Web

Sensor Web propojuje SOA a senzorovou síť, která vede k přístupu k heterogenním senzorovým zdrojům při zachování nezávislého programování. Jednou z open-source platforem pro budování Senzorového Webu je Open Sensor Web Architecture (*OSWA*). Architektura je postavena nad standardy OGC, konkrétně OGC SWE [51]. OSWA rozšiřuje SWE mimo jiné i o podporu middlewere pro budování Senzorového Webu.

Se vzrůstajícím počtem zařízení pracujících na úzkopásmových přenosech využívající nízkého výpočetního výkonu, které jsou schopny téměř v reálném čase měřit dostatečně přesně sledované jevy (např. teplota, vlhkost, poloha), vyvstává potřeba tyto informace zachytit, uložit, zpracovat a předat uživatelům. Jedním z hlavních cílů Senzorového Webu je přenést informace ze senzorů co nejjednodušeji a co nejefektivněji na server, kde budou následně zpracovány. Na obrázku 16 je znázorněna zjednodušená funkcionalita Senzorového Webu, kde jsou aktuální i historická data zpřístupněna uživateli,

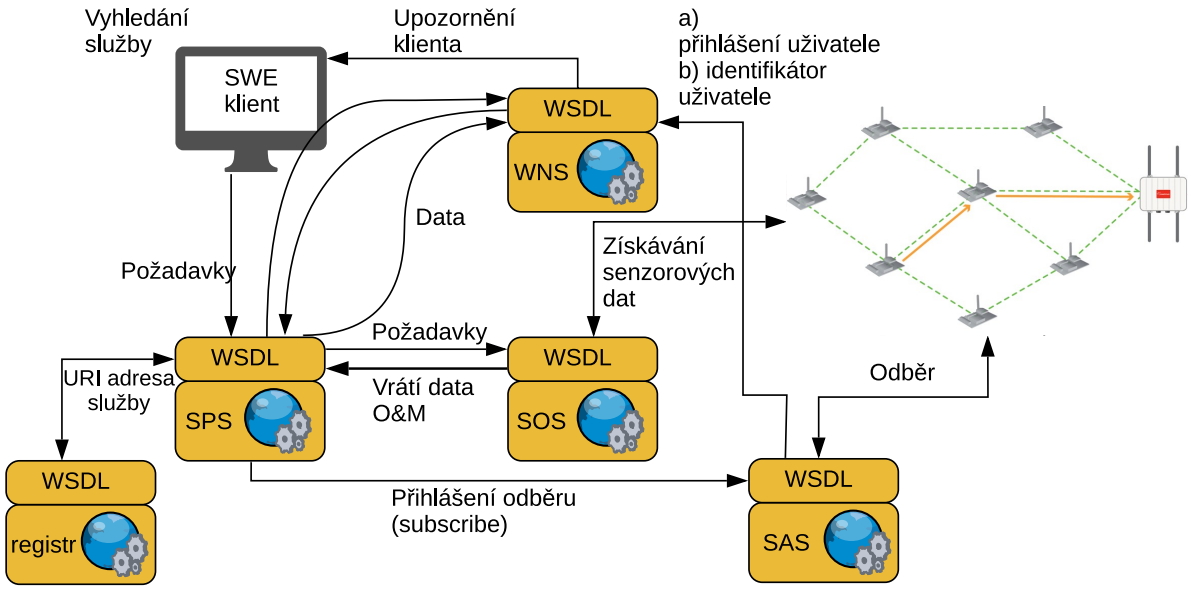
vypočítávají se statistky a vytvářejí se modely [51]. OGC definovala specifikaci SWE pro možnosti vyhledání senzorů, přístupu k senzorům, získání senzorových dat a v neposlední řadě zpracování senzorových dat. Mezi základní komponenty SWE patří:

- Sensor Model Language (*SensroML*);
- Observations and Measurements Schema (*O&M*);
- Transducer Markup Language (*TML*);
- Sensor Observations Service (*SOS*);
- Sensor Planning Service (*SPS*);
- Web Notification Service (*WNS*);
- Sensor Alert Service (*SAS*).

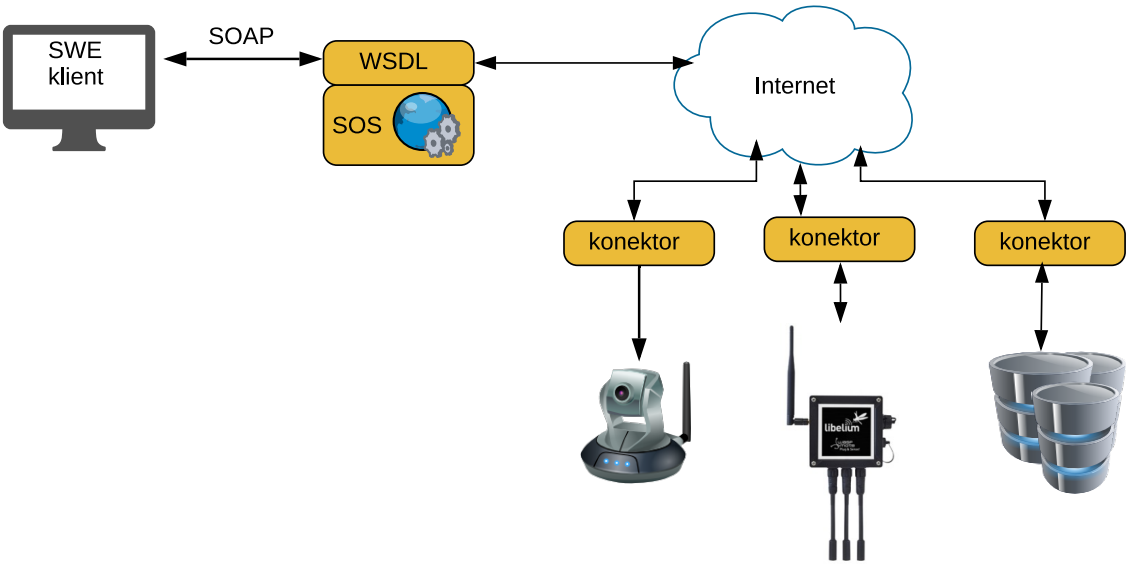


Obrázek 16: Abstraktní vize Senzorového Webu. Zdroj: [51]

Základem Senzorového Webu, stejně tak SOA je, aby jednotlivé služby pracovaly společně na vyřešení dotazu uživatele. SWE dovoluje uživateli přijímat data od fyzických senzorů. Obecný diagram (obrázek 17) popisuje základní předpoklad fungování Senzorového Webu na základě architektury orientované na služby. Klient zadá přesnou polohu senzoru, období (hodinu, den, týden...), a měřený fenomén a SWE klient poté vytvoří dotaz obsahující všechny informace a pošle ho SPS. SPS na základě proměnných v dotazu určí URI, po dotazu do registrů dostupných služeb a zvolí nejvhodnější SOS službu. Pokud klient žádá, aby ho Senzorový Web upozorňoval na překročení limitů, SPS si zapíše požadavek do SAS jako odběr (*subscription*). Poté SAS automaticky komunikuje s WNS, která informuje uživatele o překročení limitu (nebo změny dat). Jakmile je nastavený odběr v SPS, SOS služba bude posílat dotazy na senzorovou síť, aby získala naměřené údaje. Poté SPS opět pošle zprávu WNS, který doručí data SWE klientovi. SWE klient poté může data zpracovat či uchovat. Následující obrázky se vztahují k implementaci SWE pomocí SOA [82].



Obrázek 17: Akce Senzor webu podle SOA. Zdroj: [82]



Obrázek 18: Obecná architektura SOS. Zdroj: [82]

Architektura orientovaná na služby je evolucí předešlých architektur, postupným vývojem architektur od monolitických, přes klient-server, až po orientovanou na služby. Veškeré architektury se stále vyvíjejí a využívají nových technologických možností. Se vznikem Internetu (začátek 50. let 20. století) se původně aplikační architektury mění na architektury, které využívají distribuovaný Internetový prostor. Postupně vznikaly společnosti na tvorbu standardů pro komunikaci přes Internet (W3C, OASIS, WS-I). Tyto standardy se postupně implementovaly do aplikací. Takto vznikají distribuované Internetové architektury, mezi které patří i SOA. SOA využívá nejčastěji webových služeb (ale nemusí). Webové služby využívají protokolů, které komunikují na Internetu nebo intranetu (distribuované prostředí). Základem všech webových služeb je strukturovaný, strojově čitelný jazyk například v podobě JSON či XML. XML je značkovací univerzální jazyk, který poskytuje základní komunikační platformu. Standardy jako SOAP, ebXML, XAPP, využívají univerzálnost XML. Díky principům, kterými se XML řídí, jsou přenášené zprávy sémanticky a obsahově velmi bohaté [56]. Výhoda se mění v nevýhodu tehdy, pokud máme omezenou šířku komunikačního pásma (sběrnice), kde velikost zprávy může

být problém. Koncept IoT, kde již senzory nejsou pevně připojeny k napájení a k Internetu (intranetu), donutily organizace vyvinout mnohem úspornější protokoly na komunikaci se senzory. Některé aplikace budou lépe fungovat nad klasickými architekturami, např. n-vrstvá či objektově orientovaná. I tyto architektury v sobě nesou určité známky SOA, jelikož byly předchůdci a vzory pro SOA.

4.5 Proudový GIS

V dnešní době se postupně odkláníme od takzvaných pull-based (založených na táhnutí) systémů k push-based (založených na tlačení) systémům. Push-based systémy předpokládají, že provedené změny v aplikaci jsou okamžitě zobrazeny ve všech běžících instancích [157]. Jako příklad lze uvést Google Docs, Facebook, komunikátory (*instant messaging* – WhatsApp, Telegram, SMS) či událostmi řízené architektury. Pull-based systémy jsou takové systémy, které pouze odpovídají na dotaz od klienta, například architektura klient –server. Místo zpracování statických a historických dat se přechází na sekvenční zpracování v podobě proudového zpracování a proudových dat (*stream processing, streaming data*).

Proudový GIS rozšiřuje funkcionalitu běžného GIS na GIS, který čte, analyzuje a prezentuje přijatá data průběžně a neustále. Proudové zpracování poté generuje nové výstupy, kdykoliv se objeví nová data.

U komerčního produktu firmy Esri byl až do verze 10.2 přítomný nástroj Tracking Server [28]. Instaloval se samostatně k ArcGIS for Server. Tracking Server nahradil od verze 10.2 GeoEvent Server [27], jedná se téměř o totožné servery. GeoEvent Server podporuje modernější vstupní i výstupní konektory a jeho správa se přesunula do webového prostředí. GeoEvent Server je součástí serverů ArcGIS Enterprise. GeoEvent Server slouží převážně k budování webových aplikací. Zde je možné využít komunikaci pomocí WebSocket či výměny souborů. Je také možné propojit GeoEvent Server s pod-

nikovou geodatabází, která pod SDE slouží jako zdroj dat. ArcGIS Desktop ovšem pro aktualizaci dat musí odeslat požadavek na server, a ten proběhne nejdříve při posunutí či přiblížení/oddálení mapového podkladu. GeoEvent Server umožňuje nad daty provádět základní analýzy (filtrace, buffer, zjednodušení, kalkulátor polí), avšak pokročilejší správu a tvorbu složitějších analýz lze provádět až na ArcGIS Desktop či ArcGIS Pro. Výstupní konektor z GeoEvent Server lze propojit s databázovými systémy MongoDB a Hadoop či *distributed streaming platform* Apache Kafka, který nahradil RabbitMQ v předchozích verzích (ArcGIS Eneterprise 10.6). Zapojením Apache Kafka rozšířil GeoEvent Server funkcionalitu o mnoho nově příjmaných formátů a o horizontální a vertikální škálovatelnost.

U open-source produktu QGIS existuje od verze 3 funkce *Listen* (představena v říjnu 2017), která umožňuje zpracování proudových dat. QGIS obecně nabízí možnost připojit relační databázi PostgreSQL s nadstavbou PostGIS. PostgreSQL podporuje od verze 9.0 funkci *Notify* spolu s uživatelskými daty (*payload*). QGIS 3 rozšiřuje svou funkcionalitu o funkci *Listen*. Tímto QGIS doplňuje dosavadní model komunikace klient – server o *Publisher − Subscriber* a umožňuje získávání proudových dat. Jakmile PostgreSQL uloží nová data, okamžitě odešle zprávu QGIS a který automaticky aktualizuje data i mapové pole a může znovu spustit jakýkoli skript pro pokročilou časoprostorovou analýzu.

5 VÝSLEDKY

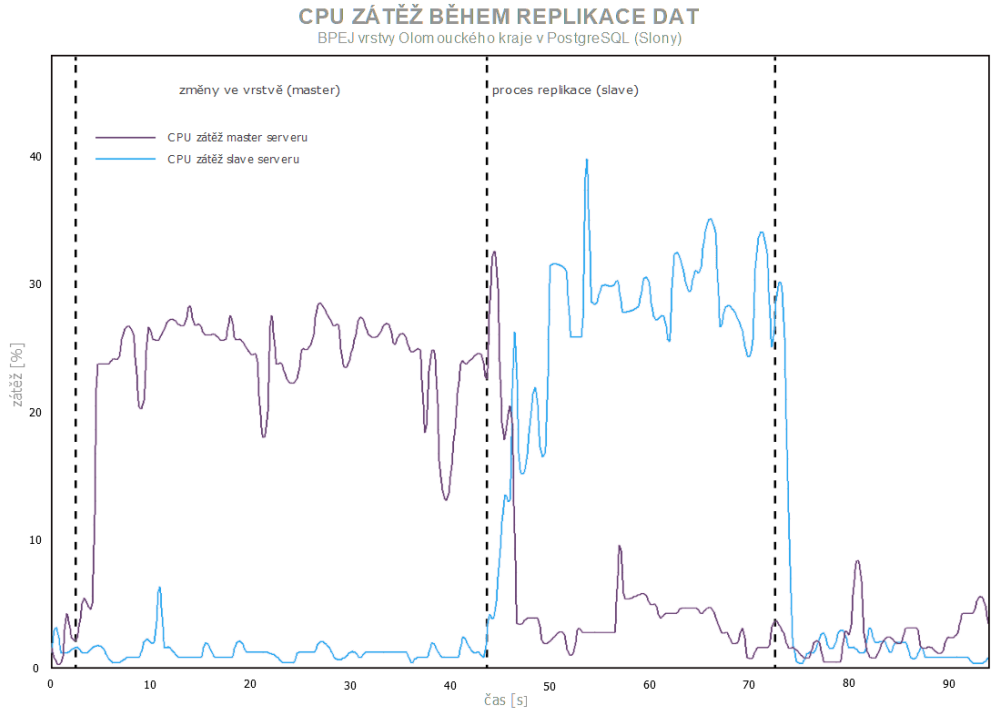
Tato kapitola popisuje získané výstupy z plnění jednotlivých dílčích cílů disertační práce. Tyto výstupy povedou k a potvrzení či zamítnutí hypotéz.

5.1 Testování replikačního mechanismu PostgreSQL a MySQL

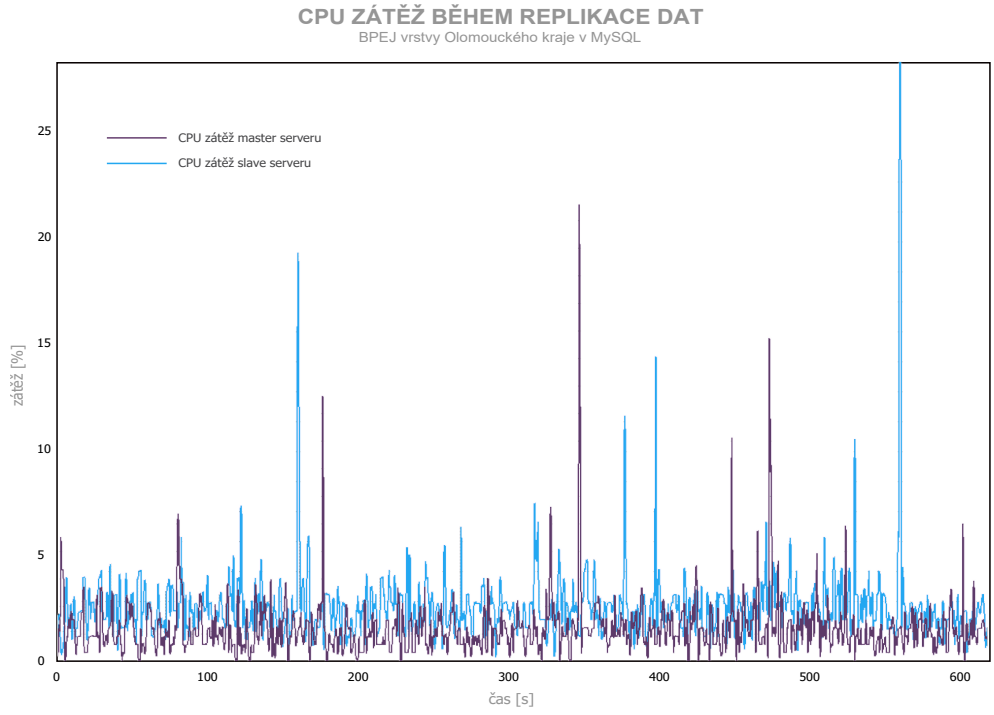
Pro hodnocení replikace prostorových dat byly testovány dva databázové servery. Jednalo se o PostgreSQL 9.5 s PostGIS 2.3.3 a MySQL 5.7.19, které byly nasazeny na serveru Katedry geoinformatiky. Testování mělo prokázat použitelnost replikačních mechanismů při správě prostorových dat a zjistit úzká hrdla (*bottleneck*) celého procesu. Jelikož úspěšnost replikace prostorových dat byla 100%, bylo měření zaměřeno na vytíženost CPU, přenosovou rychlost a celkový čas replikace jako možných úzkých hrdel celého replikačního klastru.

Vytíženost CPU

Na obrázcích 19 a 20 lze vidět vytíženost CPU (souhrn za všechny jádra CPU) během replikačního procesu BPEJ vrstvy v PostgreSQL a MySQL. Na obrázku 19 je jasně patrno, že asynchronní replikace u PostgreSQL nejprve zatěžuje *master* server a až po dokončení změn teprve posílá změny na *slave* server. Naproti tomu MySQL synchronní replikace posílá změny průběžně po jednotlivých záznamech. PostgreSQL mnohem více zatěžuje procesor než MySQL, ovšem v mnohem kratším čase. U PostgreSQL dosahuje průměrného zatížení *master* server přibližně 25 % a *slave* server 30 % zatížení během 40, respektive 30 sekund. To je především způsobeno tím, že *slave* server má výkonnostně slabší CPU, což se projevuje u obou obrázků 19 i 20. U MySQL celá synchronizace (obrázky 20, 22 a 26) trvá až 56× déle (u vrstvy BPEJ), než u PostgreSQL, ovšem s minimálními požadavky na CPU. Je možné zátěž CPU do 5 % považovat klidový stav. Výrazná lokální maxima u MySQL replikace mohou značit vyšší počet lomových bodů.

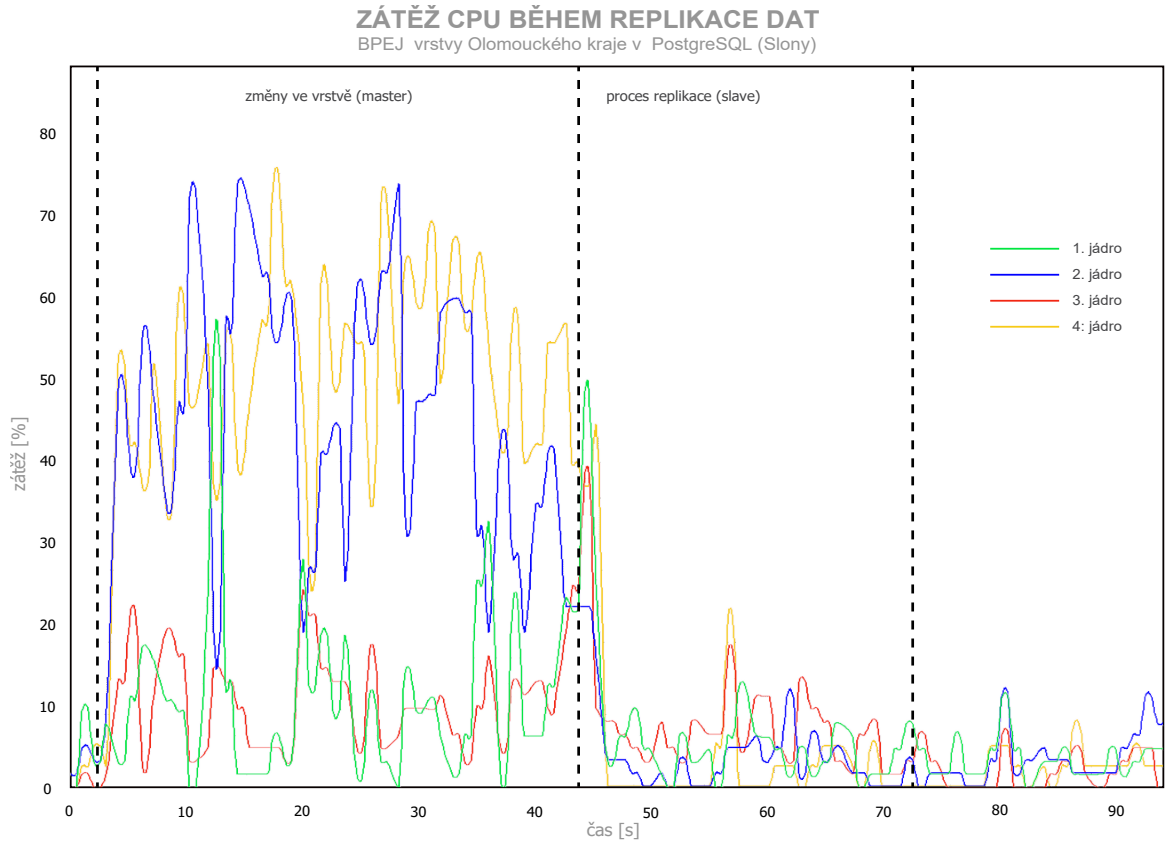


Obrázek 19: Graf vytíženosti CPU při replikaci BPEJ vrstvy v PostgreSQL (Slony)

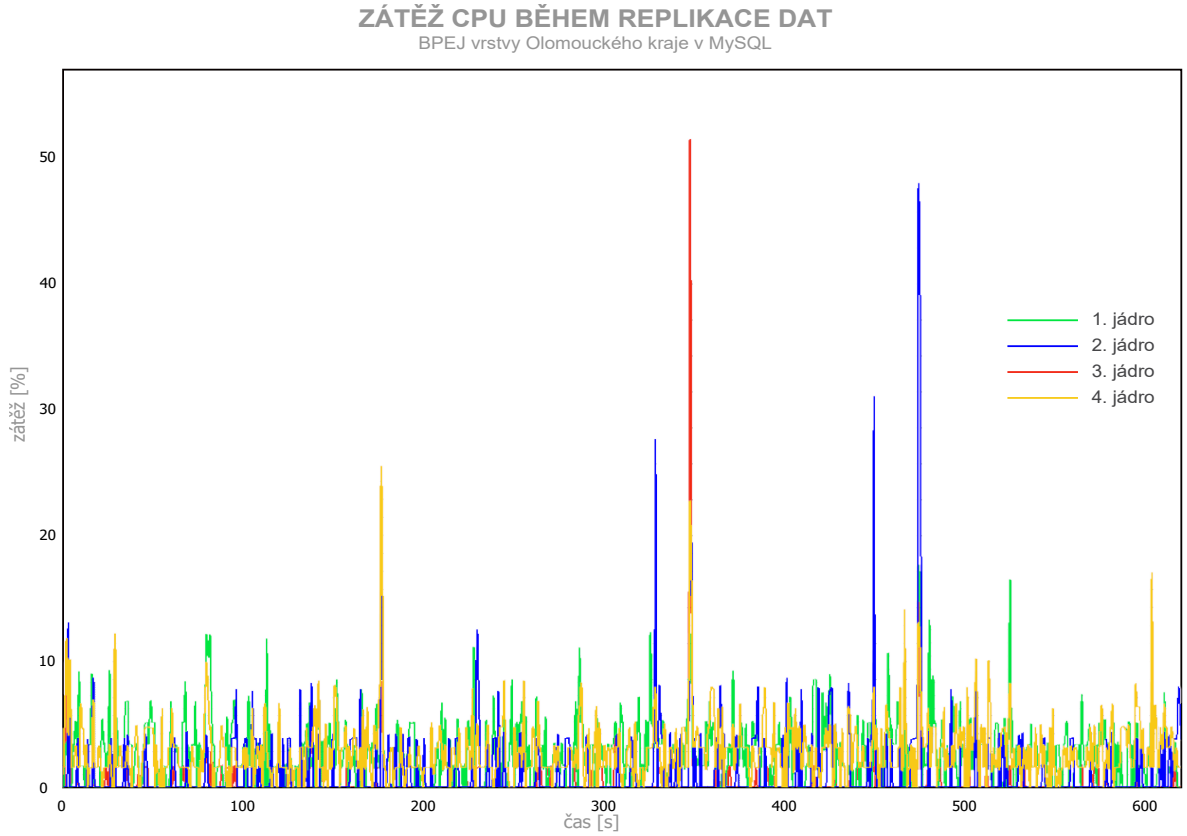


Obrázek 20: Graf vytíženosti CPU při replikaci BPEJ vrstvy v MySQL (výřez)

Na obrázcích 21 a 22 je zátěž u *master* serveru rozložena na jednotlivá jádra CPU. U PostgreSQL je v první části grafu (obrázek 21) vysoká aktivita druhého a čtvrtého jádra označena jako zpracování dat. Jádra ukládají do databáze změny provedené v software QGIS. Při replikačním procesu (druhá část grafu označena jako replikační proces) klesá aktivita všech jader na minimum. U MySQL se na ukládání a replikaci podílejí všechna jádra téměř identicky. Způsob, jak databázové servery využívají jednotlivá jádra CPU není zjistitelné.



Obrázek 21: Graf vytíženosti CPU jader master serveru při replikaci BPEJ vrstvy v PostgreSQL



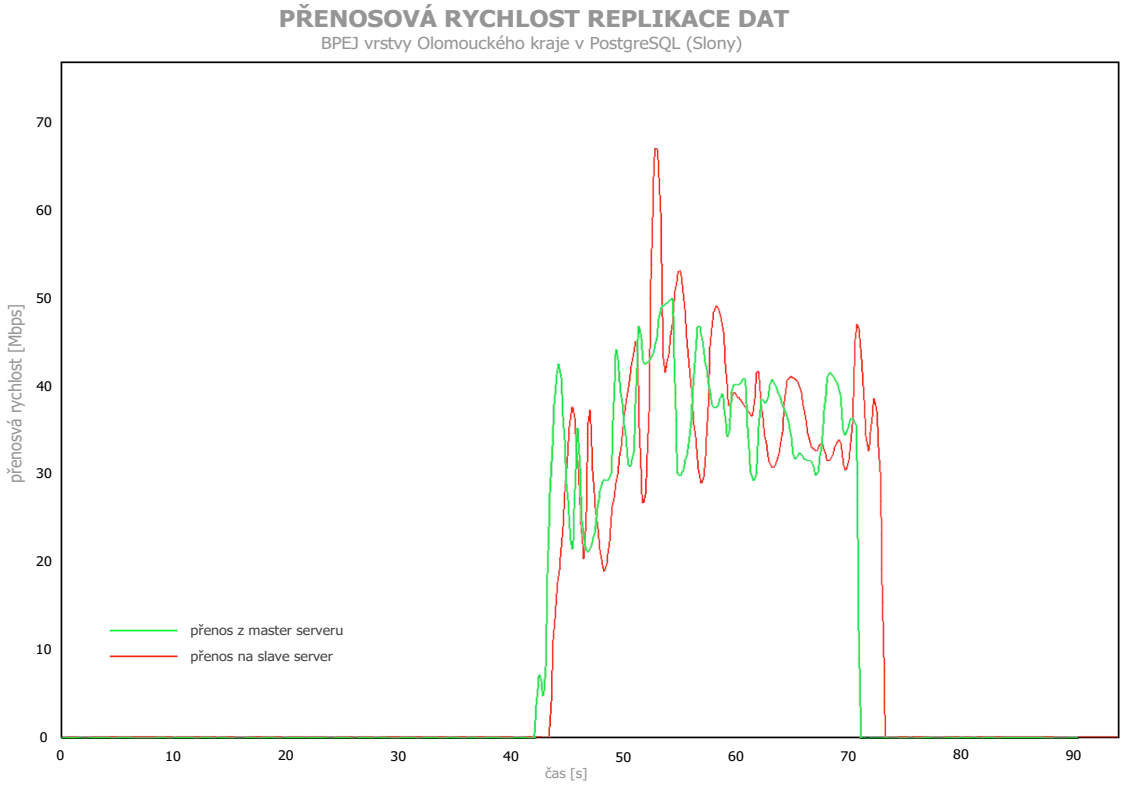
Obrázek 22: Graf vytíženosti CPU jader master serveru při replikaci BPEJ vrstvy v MySQL (výřez)

Výsledek sledování zátěže CPU při replikacích jasně ukázal, že PostgreSQL s nadstavbou Slony umožňuje rychlejší replikaci i přes její asynchronní povahu, ale vyžaduje výkonnější CPU. Naproti tomu MySQL využívá synchronní replikace, která je ovšem téměř devětkrát pomalejší, ale nezatěžuje tolik procesor. Nutné je brát zřetel na i na povahu replikovaných dat, tedy polygonová vrstva BPEJ s 3 725 023 lomovými body, 31 280 záznamy o celkové velikosti 228 MB.

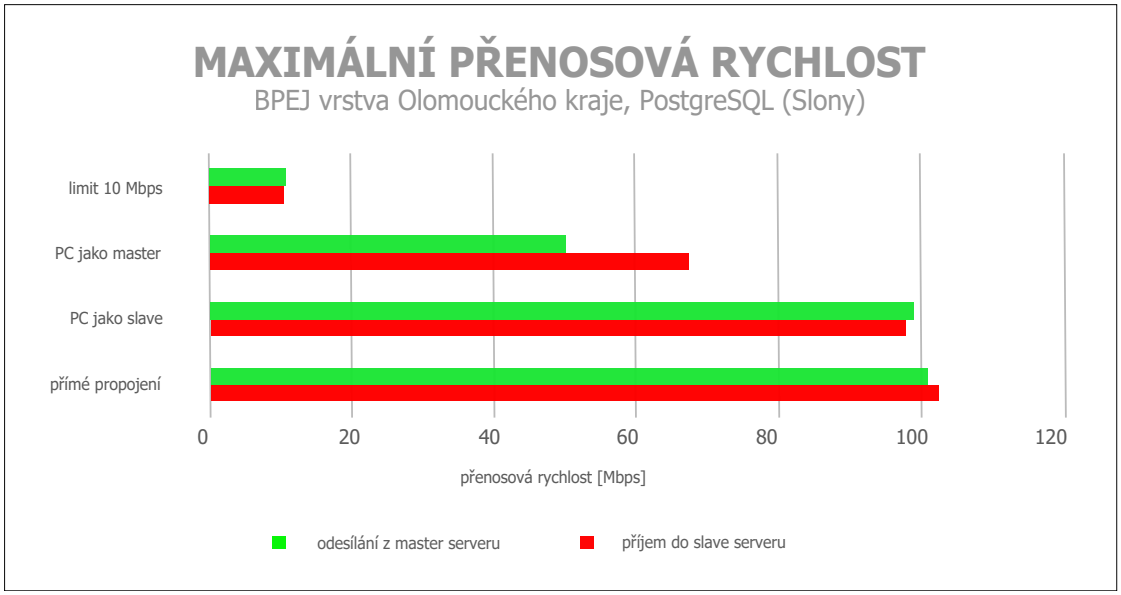
Přenosová rychlost

Dalším měřeným kritériem pro hodnocení replikačních mechanismů byla přenosová rychlost dat mezi databázemi v replikačním klastru. Následující obrázky 23 a 25 ukazují průběh rychlosti přenášených dat z PC jako *master* server do notebooku jako

slave server pro PostrgreSQL a MySQL databázi. Propojení serveru na obrázcích 23 a 25 bylo přes router se 100 Mbps LAN porty. Obecný průběh obou grafů koresponduje s využitím CPU během replikace. I zde se projevuje asynchronicita PostgreSQL a synchronicita MySQL tak, jak se projevila u vytíženosti CPU. Na obrázku 23 je vidět začátek přenosu dat z *master* serveru a o jednu sekundu později *slave* server začíná data stahovat. I zde, stejně jako při využití CPU, se PostgreSQL snaží využít maximum zdrojů serveru pro co nejrychlejší synchronizaci dat. To samozřejmě klade vyšší požadavky na samotný hardware serveru. Pokud na serveru poběží další služby, které by potřebovaly využívat síť, mohou se navzájem citelně ovlivňovat. Maximální přenosové rychlosti v dalších konfiguracích propojení serverů jsou zobrazeny na obrázku 24. PostgreSQL se při jakémkoli typu propojení snaží využít maximální dostupné zdroje. Překvapivé snížení rychlosti při zapojení přes router v konfiguraci *PC jako master* server došlo ke snížení průměrné rychlosti na asi 50 Mbps pro odesílání z *master* serveru. Je to způsobeno pomalejším zápisem *slave* serveru na disk.

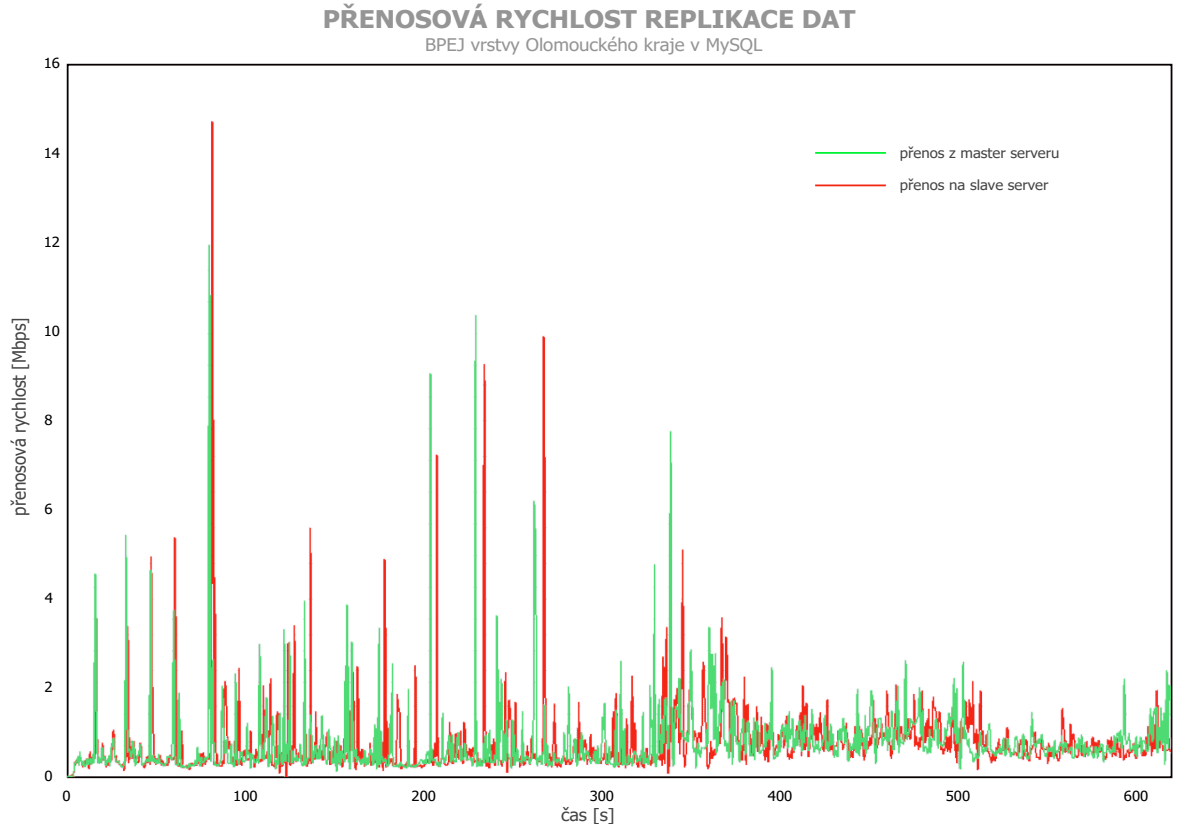


Obrázek 23: Přenosová rychlost replikace dat vrstvy BPEJ v PostgreSQL

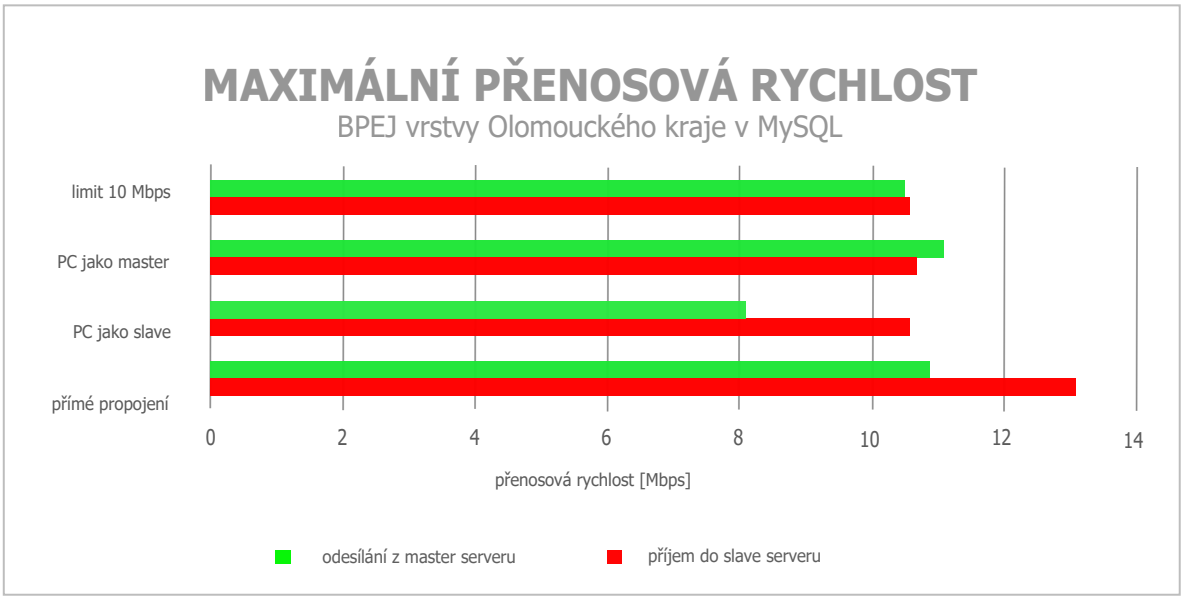


Obrázek 24: Maximální přenosová rychlost replikace dat vrstvy BPEJ v PostgreSQL

U obrázku 25 lze pozorovat, že synchronní replikace postupným odesílám jednotlivých záznamů na *slave* server vytěžuje síť jen v jednotkách Mbps. Lokální maxima v průběhu synchronizace značí přenos objemnějších dat (více lomových bodů). Maximální přenosová rychlost (obrázek 26) ukazuje, že MySQL replikace obecně těsně překročí 10 Mbps. Je to dáno odesílám jednotlivých záznamů okamžitě po jejich změně.



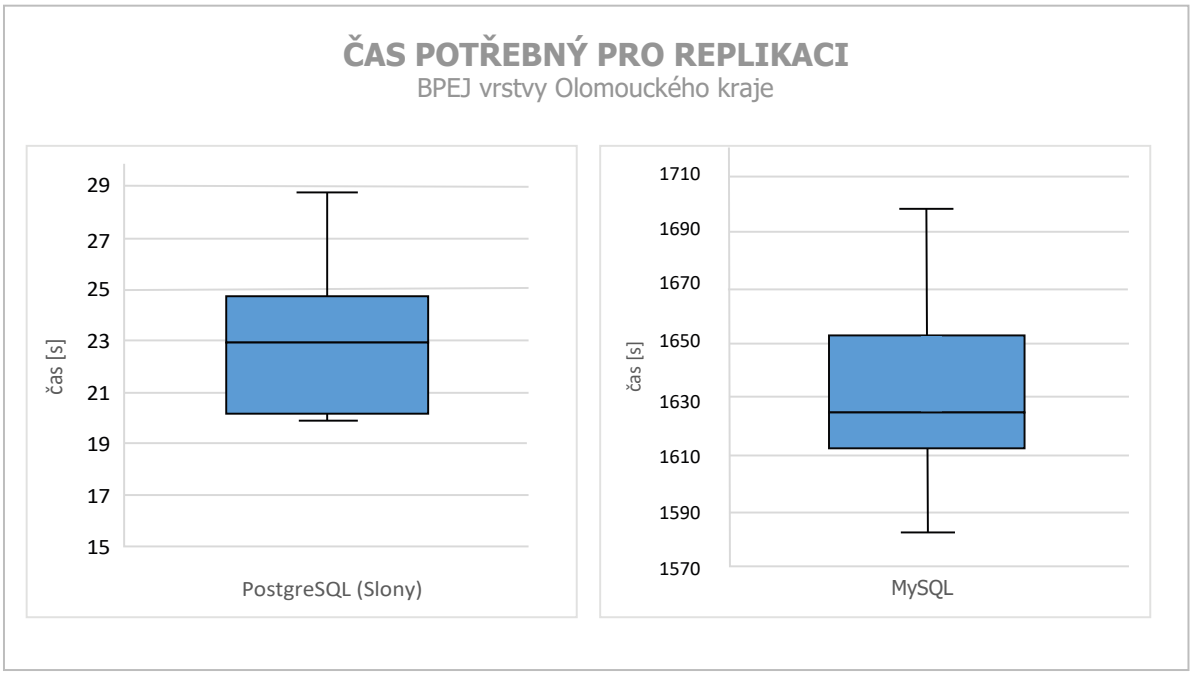
Obrázek 25: Přenosová rychlost replikace dat vrstvy BPEJ v MySQL (výřez)



Obrázek 26: Maximální přenosová rychlost replikace dat vrstvy BPEJ v MySQL

Čas replikace

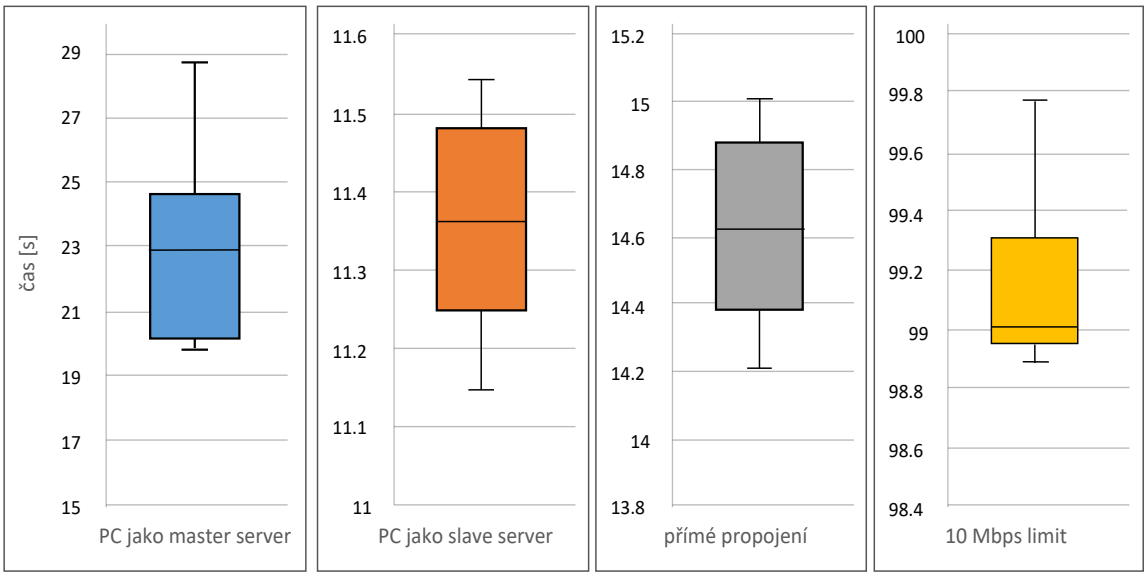
Mnoho aplikací (práce v týmu, krizový management) závisí na replikaci změn v co nejkratším čase. Následující grafy byly vytvořeny měřením času samotné replikace. Naměřené časy vychází z deseti měření každé změny vrstvy v každé konfiguraci a z výsledných časů byly vytvořeny grafy a tabulka. Již z výsledků při měření vytíženosti CPU a rychlosti přenosu dat je odvoditelné, že celkový čas replikace se bude diametrálně lišit. PostgreSQL s asynchronní replikací synchronizuje změny v BPEJ vrstvě do 29 s, MySQL stejnou operaci při synchronní replikaci zvládne do 1700 s (maximální hodnoty v obrázku 27). Hodnoty u všech boxplotů představují medián, 1. a 3. kvartil a extrémní hodnoty.



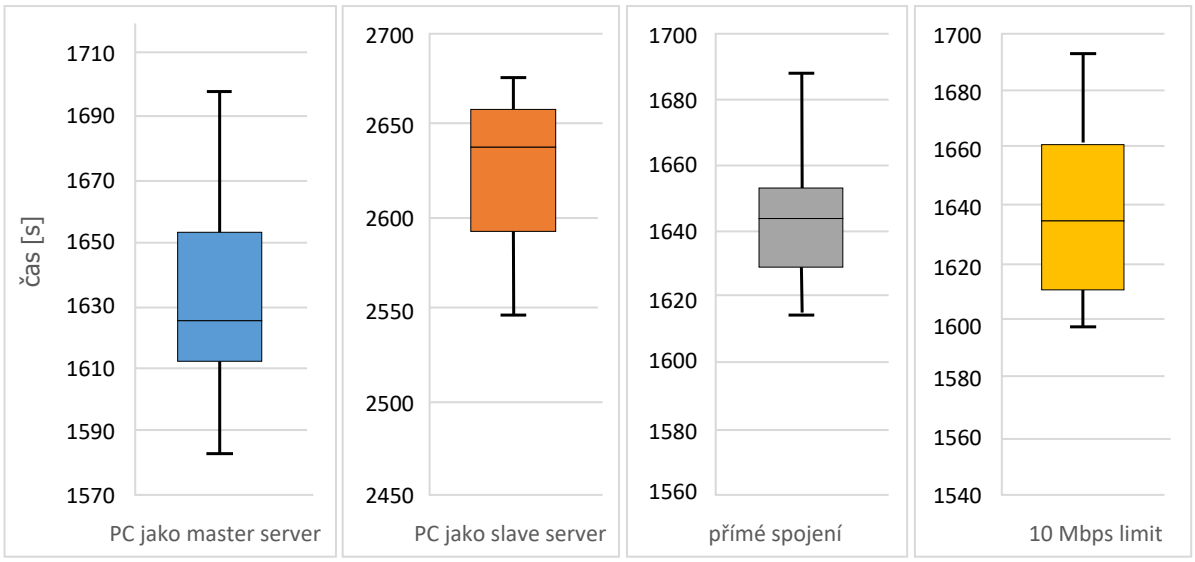
Obrázek 27: Celkový replikační čas BPEJ vrstvy v konfiguraci PC jako master server

Obrázky 28 a 29 zobrazují detailní časy synchronizací vrstvy BPEJ v různých konfiguracích propojení klastru PostgreSQL. Obecné srovnání lze vidět na obrázku 30 a 31, kde jsou zobrazeny průměrné hodnoty všech vrstev ve všech konfiguracích zapojení klastru. Detailní statistiky jsou pak uvedeny v tabulce 5 na straně 84.

Stabilita replikačních mechanismů byla potvrzena i tím, že všechny časy měly směrodatnou odchylku do 5 % od průměrného času. Pouze u PostgreSQL klastru při konfiguraci *PC jako master* server byla směrodatná odchylka 12 %.



Obrázek 28: Replikační čas BPEJ vrstvy pro různé konfigurace propojení PostgreSQL

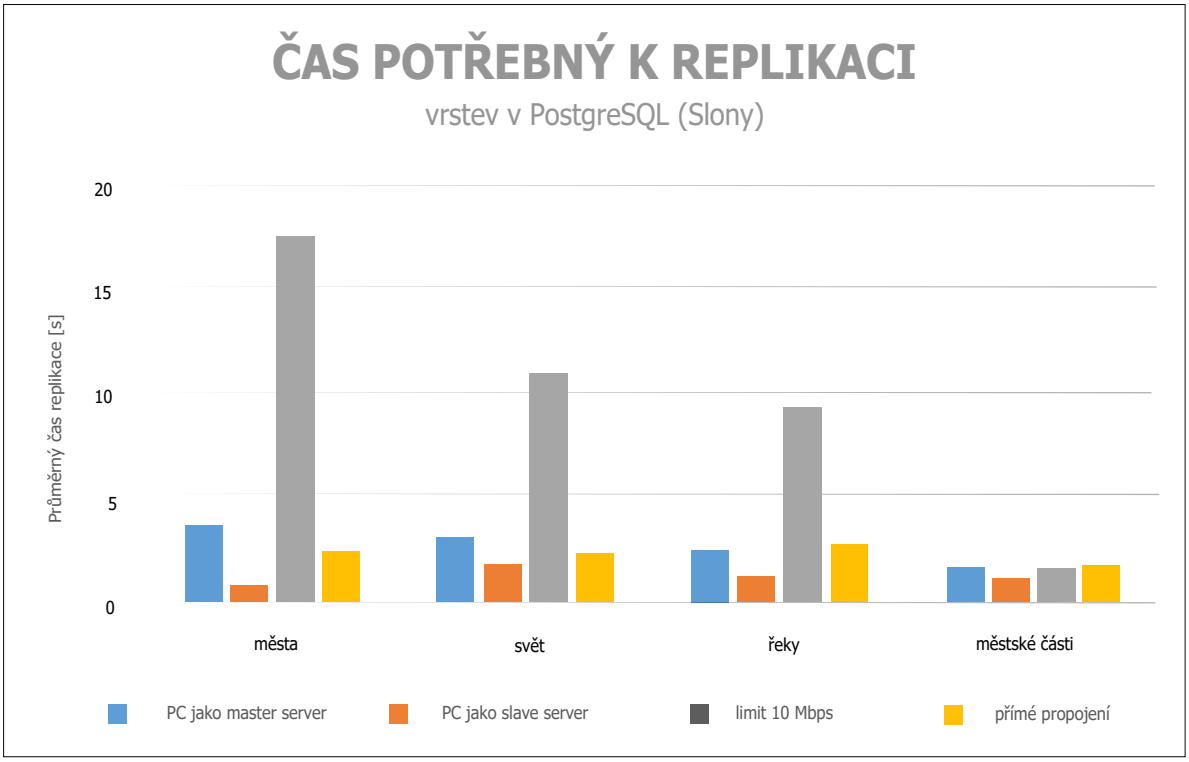


Obrázek 29: Replikační čas BPEJ vrstvy pro různé konfigurace propojení MySQL

Tabulka 5: Statistická analýza naměřených replikačních času BPEJ vrstvy [s]

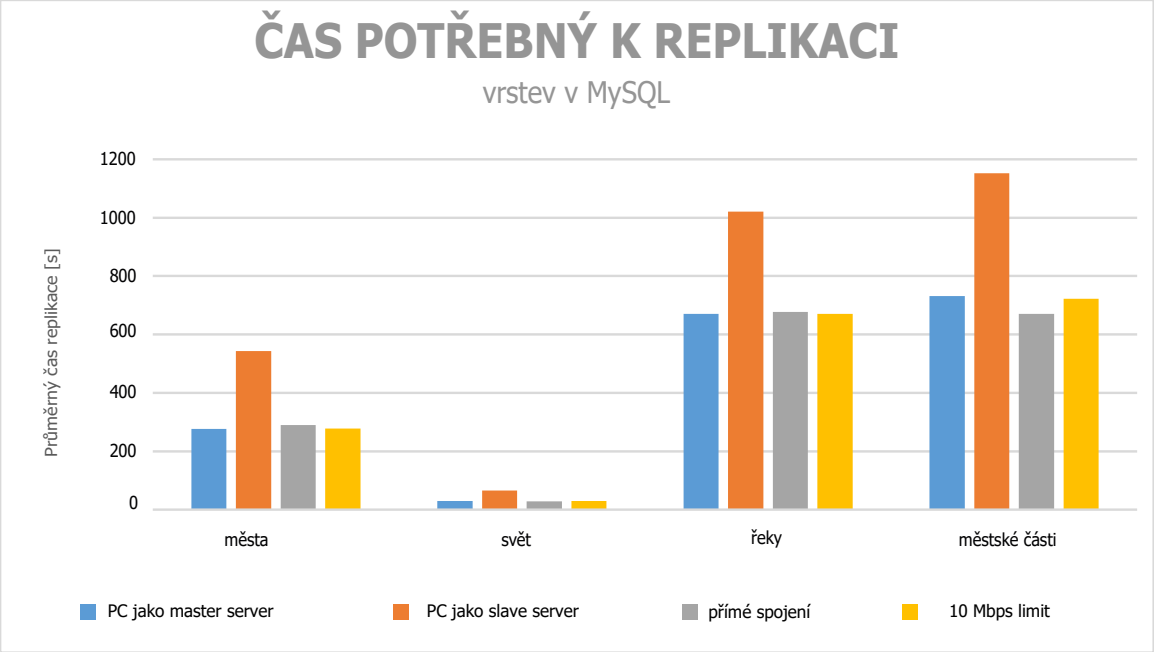
DBMS	PostgreSQL (Slony)				MySQL			
Konfigurace	PC master	PC slave	Přímé	10 Mbps	PC master	PC slave	Přímé	10 Mbps
Průměr	23,00	11,40	14,63	99,14	1633,30	2656,50	1643,70	1628,10
Rozptyl	7,39	0,05	0,07	0,07	926,01	6668,25	370,41	2451,89
Intervalové rozpětí	8,91	0,82	0,80	0,89	115,00	303,00	73,00	191,00
Směrodatná odchylka	2,72	0,22	0,26	0,26	30,43	81,66	19,25	49,52

Průměrné replikační časy u PostgreSQL klastru ukazují nejen silnou závislost na dostupné rychlosti připojení, ale také, že s vyšším počtem lomových bodů roste samotná náročnost replikačního procesu. Při konfiguraci s omezenou rychlostí přenosu na 10 Mbps je tento vliv nejvýraznější. Z tabulky 2 lze vyčíst, že polygonová vrstva světa s jediným záznamem, ale s 411 132 lomovými body, je náročnější na replikaci než o poznání objemnější bodové městské části (4 MB vs 12 MB). I při konfiguraci klastru *PC jako master* server lze toto tvrzení potvrdit. Při konfiguraci *PC jako slave* server a při přímém propojení se vyskytují abnormality. Například, že městské části byly synchronizovány rychleji při omezené rychlosti než při přímém zapojení, i když jen o 0,02 s (1,1 %).



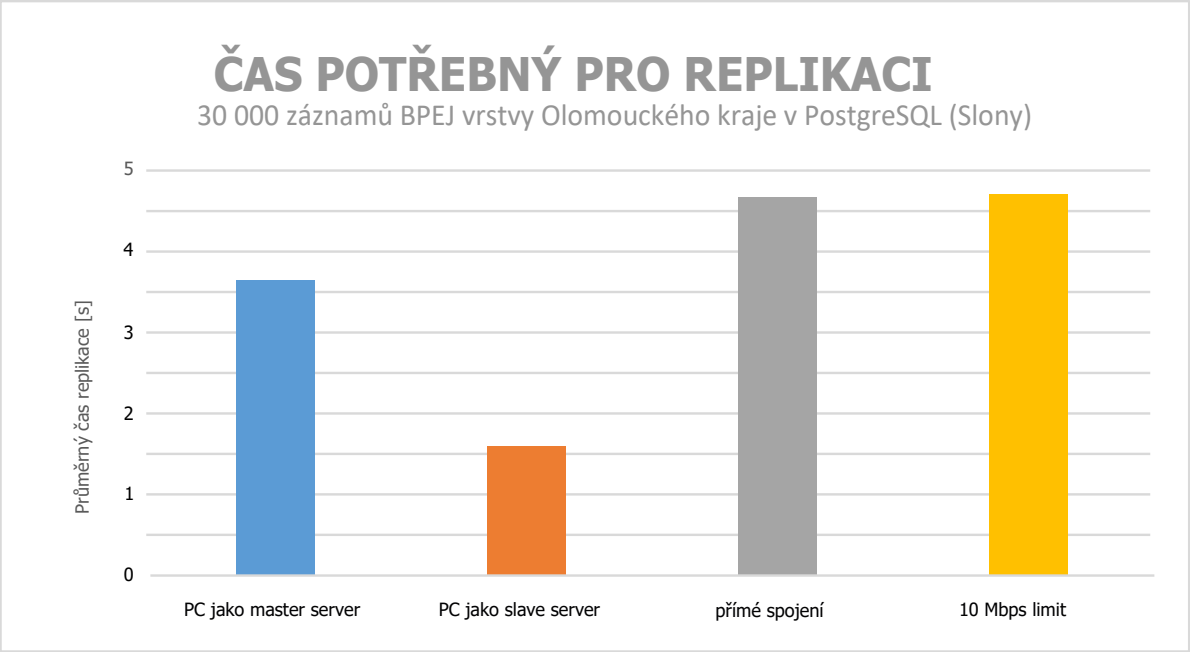
Obrázek 30: Průměrné replikační časy vrstev v různé konfiguraci klastru v PostgreSQL

U MySQL replikace graf průměrného replikačního času (obrázek 31) naopak ukazuje, že nejvíce ovlivňujícím faktorem je počet synchronizovaných záznamů. Až na konfiguraci s přímým propojením u vrstvy městských částí všechny časy odpovídají tomu, že s vyšším počtem záznamů roste časová náročnost samotné replikace. Řeky a městské části mají podle tabulky 2 na straně 8 velmi podobný počet záznamů, a naprosto rozdílný počet lomových bodu (body vs. linie) a i přesto byly časy velmi podobné (rozdíl při přímém propojení je menší než 1 %). Navíc se ještě projevuje závislost na výkonu *master* serveru, kdy výměna rolí *PC jako master* na *PC jako slave* server měla čas od 45 % až po 65 % horší, než když byl *PC master server*.



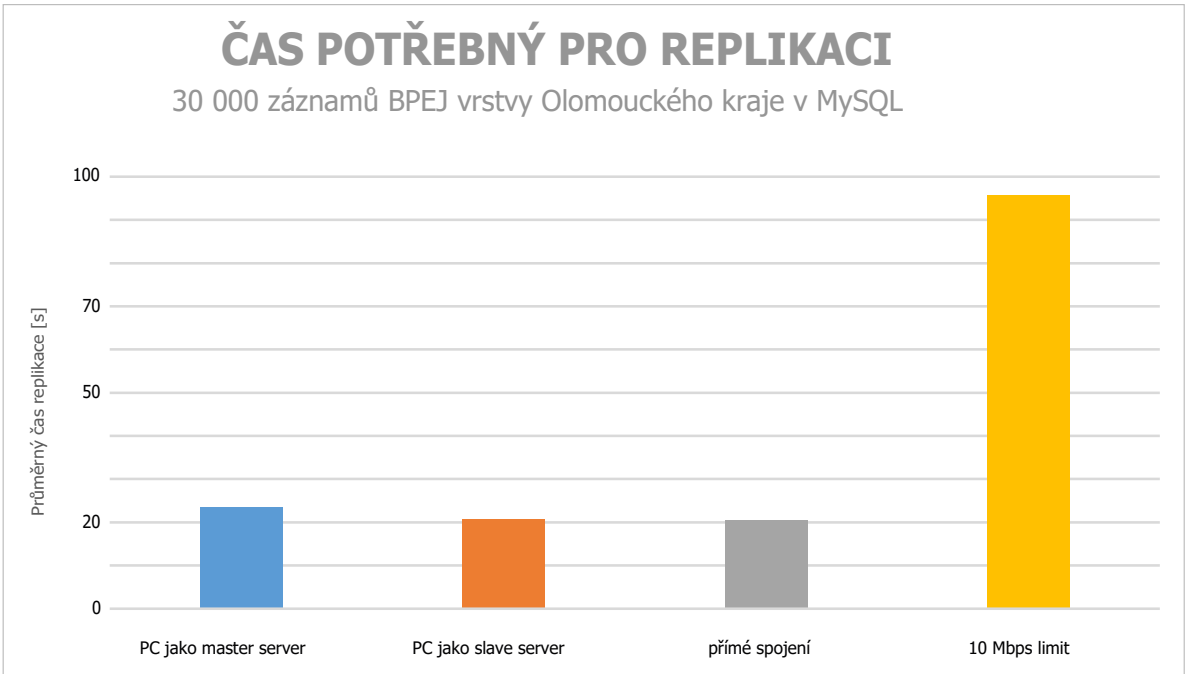
Obrázek 31: Průměrné replikační časy vrstev v různé konfiguraci klastru v MySQL

Testování replikačních časů u negeometrického atributu je další běžnou úlohou při práci s prostorovými daty. Pro zajištění dostatečně dlouhé doby replikace bylo upraveno 30 000 záznamů ve vrstvě BPEJ. I tak změna trvala u PostgreSQL do 5 s (obrázek 32) a u MySQL do 100 s (obrázek 33). Zde se také ukazuje vliv velikosti negeometrického a geometrického atributu, kdy negeometrický atribut by při správném návrhu databázového schématu měl respektovat minimálně 3. normální formu. U PostgreSQL (obrázek 32) se i u negeometrického atributu projevuje závislost výkonu *slave* serveru na celkový čas replikace. Omezení rychlosti v tomto případě nemělo žádný vliv na přenos replikace, i když bylo s přímým propojením nejpomalejší.



Obrázek 32: Průměrný čas replikace atributu u BPEJ vrstvy v různých konfiguracích klastru PostgreSQL

U MySQL se naopak poprvé projevilo omezení rychlosti přenosu na 10 Mbps, které mělo za následek trojnásobné prodloužení času. Dalo by se usuzovat, že MySQL replikace je více optimalizovaná na negeometrické replikace. Zmizel i vliv výkonnějšího zařízení jako *master* server. I tak byl MySQL více než čtyřikrát pomalejší při replikaci než PostgreSQL.



Obrázek 33: Průměrný čas replikace atributu u BPEJ vrstvy v různých konfiguracích klastru MySQL

Testování mělo prokázat použitelnost a využitelnost replikačních mechanismů, jejich úskalí (*bottleneck*) při replikaci prostorových dat. Replikační mechanismy jsou primárně zaměřeny na práci s textovými a numerickými daty. Tomu odpovídají i počty dostupných testů a publikací. Práce s prostorovými daty je více specifická a mnohdy vyžaduje i specifický přístup. Replikační mechanismy u PostgreSQL s nadstavbou Slony (asynchronní logická replikace) přináší řadu výhod oproti MySQL. Například Slony dovoluje replikaci jen určitých tabulek (vrstev). Replikace je dokončena za mnohem kratší čas, který závisí na výkonosti *slave* serveru a počtu lomových bodů v replikovaných datech. Naproti tomu MySQL nezatěžuje tolik CPU a síťové připojení, čas replikace nejvíce ovlivňuje počet replikovaných záznamů, a je důležitější mít výkonnější *master* server.

Splněním prvního dílčího cíle je položen základ architektury distribuované prostorové senzorové databázové sítě. Ta umožňuje zápis, analýzu a poskytování proudových senzorových dat ze senzorových sítí využívající různé komunikační technologie webovým a GIS klientům.

Současně s výše uvedenými výsledky byl replikační mechanismus prakticky využit při řešení v konkrétních výzkumných úlohách [112, 152, 57].

Důvodem pro nasazení replikace pro výše uvedené výzkumné úlohy je práce s rozsáhlou prostorovou sadu a byla snaha využít potenciálu distribuované databázové sítě pro rozložení zátěže mezi primární databázi, kde dochází k průběžné aktualizaci datové vrstvy (na počítači doc. Pechance) a produkční databázi (počítač autora práce), která sloužila jako úložiště pro řešení analytické úlohy [122].

Předmětem replikace byla kombinovaná vrstva biotopů ČR. Jedná se o polygonovou vrstvu v rozsahu celé ČR, která v měřítku 1 : 10 000 zaznamenává výskyt a stav jednotlivých biotopů. Vrstva rozlišuje 216 typů biotopů (186 přírodních a 28 nepřírodních) a celkově obsahuje 2 085 099 segmentů (část v tabulce 6). Součástí vrstvy je sada koeficientů pro hodnocení více jak 15 vlastností (ekologická hodnota, ekonomická cena, míry plnění vybraných ekosystémových funkcí a služeb). Vlastníkem vrstvy je Ústav výzkumu globální změny Akademie Věd České republiky, v.v.i, který na vrstvě spolupracuje s Katedrou geoinformatiky. Ve vrstvě jsou využívány data od Agentury ochrany přírody a krajiny ČR. Vrstva se využívá k integrovanému modelování a hodnocení stavu krajiny a míry plnění ekosystémových funkcí a služeb.

Tabulka 6: Ukázka části atributové tabulky s expertními znalostmi

Kód	Biotop	Land-cover	Biotop									...	Ekosystémová funkce			
			Zralost	Přirozenost	Strukturní diversita	Druhová diversita	Vzácnost biotopu	Vzácnost druhů	Zranitelnost	Ohroženost	Ekologický bod		Evapotranspirace $[litr/m^2/rok]$	Sekvestrace uhlíku – nadzemní biomasa $(tuny\ C/ha)$	Sekvestrace uhlíku – podzemní biomasa $(tuny\ C/ha)$	Sekvestrace uhlíku – mrtvá biomasa $(tuny\ C/ha)$
L3.1	Hercynian oak-hornbeams	listnatý les	4	6	6	5	3	3	3	4	47	...	700	97	21,3	9,05
L3.2	Polonian oak-hornbeams	listnatý les	4	6	6	5	5	3	3	4	55	...	700	97	21,3	9,05
L3.3	Pannonian-Carpathian oak-hornbeam	listnatý les	4	6	6	5	5	4	3	4	58	...	700	97	21,3	9,05
L3.4	Pannonian oak-hornbeam	listnatý les	4	6	6	6	5	4	3	4	61	...	700	97	21,3	9,05

5.2 Ověřený postup zpracování dat ze senzorové sítě

Tato kapitola popisuje výstupy získané v rámci řešení druhé dílčího cíle, který se zabývá přenosem dat ze senzorů pomocí LPWAN sítí a zpracováním takto získaných dat.

5.2.1 Automatizované zpracování dat do podoby umožňující bezprostřední využití v analytických úlohách

Hlavním výsledkem tohoto dílčího cíle bylo vytvoření funkčního a ověřeného automatizovaného zpracování senzorových dat do podoby, která umožňuje bezprostřední využití v analytických úlohách (*preprocessing*). Předmětem řešení bylo stávající „senzorové vybavení" Katedry geoinformatiky [69]. Aktuální portfólio senzorů a senzorových sítí zahrnuje několik odlišných typů čidel, senzorových desek a podporovaných komunikačních rozhraní. Podrobné informace jsou v tabulce 7, která má návaznost na obrázek 35 na straně 101 a na obrázek 38 na straně 113.

Tabulka 7: Seznam čidel a senzorů

Čidlo	Senzor	Označení v jednotné databázi	Komunikace
Sensiron SHT75	Libelium	HUMB, TCB	Sigfox, LoRaWAN, Zigbee
AMAT 200 mm^2	Libelium, EasyLogGSM	PLV1, PLV2	Sigfox, LoRaWAN, Zigbee, GPRS
Decagon EC-5	Libelium, EasyLogGSM	SOIL1, SOIL2	Sigfox, LoRaWAN, Zigbee, GPRS
DFR0300	Libelium	EC	Sigfox, LoRaWAN, Zigbee
DS18B20	Libelium	TCA	Sigfox, LoRaWAN, Zigbee
HumiAir9	EasyLogGSM	HUMB, TCB	GPRS
Fielder SR03 - 500 mm^2	EasyLogGSM	PLV1	GPRS
Fielder SR02 - 200 mm^2	EasyLogGSM	PLV1	GPRS
Fiedler Virrib	EasyLogGSM	Virrib	GPRS
Baterie	Libelium, EasyLogGSM	BAT	Sigfox, LoRaWAN, Zigbee, GPRS

Tato diverzita jednotlivých řešení a podporovaných komunikačních protokolů způsobuje vysoké nároky na (před)předzpracování vstupních (senzorových) dat při každé úloze. Nutí uživatele mít neustále všechny informace o daném senzoru, způsobu komunikace a o formátu generovaných dat, stejně jako aktivní znalost a neustálý přehled k několika proprietárním programovým řešením pro stažení a kontrolu vstupních dat. Každé technologické řešení rovněž generuje odlišné systémové a nahodilé chyby. Při stávajícím rozsahu senzorové sítě a rychlosti (a objemu) produkce primárních dat je ruční a dílčí *preprocessing* při každém zpracování dále neudržitelné. Předkládané řešení tyto nedostatky odstraňuje a nabízí všem analytikům na pracovišti okamžitý přístup k očištěným a „sjednoceným“ datům.

Dílčí část tohoto postupu byla certifikována v podobě certifikované metodiky č. UKZUZ 020993/2017 Integrace obrazových materiálů s daty ze senzorové sítě [113], na které autor spolupracoval. Dále uvedené postupy jsou snadno přenositelné a zobecnitelné na typové řešení např. podle výrobce či komunikačního protokolu.

Následuje popis jednotlivých dílčích kroků zpracovaní dat podle skupin senzorů v závislosti na využívané komunikační technologií.

Senzory komunikující pomocí Sigfox

Senzory komunikující pomocí Sigfox a LoRaWAN technologie posílají data v hexadecimálním zápisu, kde každá hodnota je dvoumístná jednobytová hexadecimální hodnota (algoritmus 2). Sigfox umožňuje přenášet maximálně 12 B dat v jedné zprávě a maximálně 140× za den (detailnější popis v kapitole 4.3.1.3). Hodnoty měřené senzorem, které obsahovali desetinná čísla, bylo nutno před odesláním upravit do podoby, která by usnadňovala přenos Sigfox a LoRaWAN technologií. U hodnot kapacita baterie v procentech (proměnná *bat* v ukázkovém algoritmu 2), relativní vlhkost s_hum, půdní vlhkosti *ec51* a *ec52* se mohou vyskytovat jen hodnoty od 0 do 100, v hexadecimálním zápisu pak 0 až 64. Hodnoty srážkoměru (*pluv* a $pluv\_last\_hour$), které vyjadřují počet

impulzů v dané hodině a v předešlé hodině, jsou celými čísly. Ani zde se nepředpokládá, že by hodnota srážek byla nad 255 impulzů za hodinu. Člunek u srážkoměru se záchytnou plochou $200mm^2$ má kapacitu 4 ml na jedno překlopení, které odpovídá $0,2mm$ srážek. V tomto nastavení lze odeslat počet překlopení člunku do úhrnu srážek $50mm$ za 1 hodinu. Pokud by se očekávala vyšší srážka, musí se odesílaná data rozšířit o další byte. U sledování teplot, které mohou nabývat hodnot od -40 do $+80$ (podle provozních hodnot čidla) je nutné odstranit záporná čísla, která by zbytečně zvětšila velikost dat. Ke změřené teplotě se proto přičte hodnota 50. Tím se budou přenášet data z rozsahu od 0 do 100 při teplotách od -50 do $+50$. Měření teploty je navíc poměrně citlivé (odchylka do 3 % podle čidla), a proto se odesílají i desetiny stupně (*rest_ decimal*). Jelikož senzor obsahuje dvě teplotní čidla, je využit rozsah 1 B dat tím způsobem, že hodnota desítek (v decimálním zápisu) je určena jednomu čidlu a hodnota jednotek je určena druhému čidlu. Proměnná *rest_ decimal* bude tedy nabývat hodnot od 00 do 99 (v decimálním zápisu). Senzor Libelium odesílají v příkladu algoritmu 2 v technologií Sigfox 10 B dat, každý 15 minut.

Algoritmus 2 Odeslání naměřených dat pomocí Sigfox technologie

```
// odeslání naměřených dat
snprintf( data, sizeof(data), \
        "%02x%02x%02x%02x%02x%02x%02x%02x%02x%02x", \
        bat, s_temp, s_hum ,ec51, ec52, ec, ec_temp, \
        rest_decimal, pluv, pluv_last_hour);
Sigfox.send(data);
```

Registrace Sigfox zařízení probíhá na základě přidělených kódů ke každému komunikačnímu zařízení. V tomto případě se jedná o Sigfox modul pro Arduino, Waspmote a Raspberry Pi. Každý tento modul má vlastní identifikační kód a tzv. PAC (*Porting Authorization Code*). Oba tyto kódy slouží k identifikaci jednoho modulu. Tyto údaje jsou dodány spolu s modulem. Sigfox modul poté komunikací s bránou vysílá spolu

s maximem 12 B uživatelských dat i identifikátor, který pak síťový server propojí s konkrétním uživatelským účtem v Sigfox Cloud (obrázek 11 na straně 51). Spárování Sigfox modulu s Sigfox Cloud poté probíhá čistě přes odeslání první testovací zprávy. Následně je pak uživateli nabídnuto několik způsobů, jak získat data ze Sigfox Cloud. Data lze získat pomocí REST API a Callback API. U REST API po zadání URL adresy a přihlášení, lze získat data ve formátu JSON či YAML. REST API je omezeno na jeden dotaz za sekundu, je proto pro získávání proudových dat nevhodné, ať již z hlediska omezeného počtu dotazů, tak i samotným faktem, že pro získání dat je potřeba odeslání dotazu od klienta. Ten může přijít v době, kdy žádná nová data nebyla přijata, nebo budou přijata těstě po požadavku. U jednoho zařízení lze jednosekundový limit připustit, ovšem u 60 zařízení již dotaz na jedno zařízení bude jen každou minutu. REST API je určeno spíše k administrátorské správě a jednorázovému získání informací o zařízení.

Oproti tomu získávání dat pomocí Callback, které přeposílají data ihned po jejich přijetí (událostmi řízená data, proudová data). Sigfox nabízí jak uživatelský, tak přednastavený Callback. Uživatelský Callback lze nastavit, aby odesílal informace o datech, servisních informací, chybách či událostech. Callback dat může být navíc nastaven tak, aby pouze přeposílal přijatá data na další server nebo může po přeposlání zprávy čekat na data od serveru, která následně pošle do senzoru. Odesílat data do senzoru lze ovšem jen čtyřikrát denně a senzor na ně musí aktivně čekat.

Samotné odesílání dat ze Sigfox Cloud lze na webový server pomocí metod GET, POST, PUT, nebo pomocí emailu. Jako nejpraktičtější se ukázalo odesílání dat pomocí metody POST na webový server. Webový server je hostovaný na platformě Microsoft Azure, který nabízí bezplatný hosting pro Python Web Server (více v kapitole 5.3 na straně 100). Sigfox Cloud odesílá data na webový server pomocí šifrovaného propojení ve formátu JSON.

Struktura dat (reprezentovaných pomocí JSON) je nakonfigurována přímo v Sigfox Cloud (algoritmus 4) a nabízí předefinované hodnoty, jako například datum, název zařízení či přijatá originální data. Originální data lze přímo v Sigfox Cloudu rozkódovat pomocí uživatelské konfigurace (algoritmus 3). Ta udává počet bitů a v jaké posloupnosti se ve zprávě vyskytují požadované hodnoty. Tím, že senzor je naprogramován tak, aby odesílal vždy dvoumístné celočíselné číslo (*unsigned integer*) jsou přijatá data v hexadecimálním zápisu rozkódována pomocí následující konfigurace (algoritmus 3), kde *bat* je název proměnné v rámci Sigfox Cloudu a *uint:8* představuje osmi-bitové celé číslo bez znaménka. Po této uživatelské definici přijatých dat lze vytvořit JSON zápis přímo s rozkódovanými uživatelskými daty (algoritmus 4).

Algoritmus 3 Uživatelské nastavení rozkódování hexadecimálních dat přijaté od senzoru v Sigfox Cloud

```
bat::uint:8 s_temp::uint:8 s_hum::uint:8 ec5_1::uint:8
ec5_2::uint:8 ec::uint:8 ec_temp::uint:8 rest::uint:8
pluv::uint:8 pluv_prev_hour::uint:8
```

Algoritmus 4 JSON odesílaný ze Sigfox Cloud na webový server

```
{
        "device":"{device}",
        "time":"{time}",
        "data":"{data}",
        "bat":{customData#bat},
        "s_temp":{customData#s_temp},
        "s_hum":{customData#s_hum},
        "ec5_1":{customData#ec5_1},
        "ec5_2":{customData#ec5_2},
        "ec":{customData#ec},
        "ec_temp":{customData#ec_temp},
        "pluv":{customData#pluv},
        "pluv_prev_hour":{customData#pluv_prev_hour}
}
```

Senzory komunikující pomocí LoRaWAN

LoRaWAN komunikační technologie je po stránce administrace velmi podobná administraci Sigfox. Pro LoRaWAN existuje oficiální poskytovatel (CRa) i soukromí poskytovatelé. Tito soukromí poskytovatelé vlastní LoRaWAN bránu, nejčastěji se jedná o Raspberry Pi s nadstavbou. Tato brána následně přeposílá data cloudovému poskytovateli, například The Things Network, která nabízí propojení soukromých bran do celosvětové sítě tak, aby je mohl zdarma využívat každý uživatel. CRa využívají platformy od firmy Loriot. Největší rozdíl mezi Sigfox a LoRaWAN od CRa je ten, že u LoRaWAN nelze nadefinovat vlastní formát JSON (algoritmus 5), který bude přeposílán na server. Nejdůležitější hodnoty jsou *\"ts\"* - čas přijetí a *\"data\"* - hexadecimální zápis dat odeslaných senzorem.

Algoritmus 5 LoRaWAN JSON

```
Content-Type: application/json
{"type":"D","data":"{\"cmd\":\"gw\",\"EUI\":\"XXXXXXXXXXXXXXXX
    \",\"ts\": 1486970750794,\"fcnt\":4368,\"port\":1,\"freq
    \":867700000,\"toa\":1482,\ "dr\":\"SF12 BW125 4/5\",\"ack
    \":false,\"gws\":[{\"rssi\":-120,\"snr\":- 6.5,\"ts
    \":1486970750794,\"gweui\":\"024B0BFFFF03054B\"},{\"rssi\":-
     121,\"snr\":0.2,\"ts\":1486970750825,\"gweui\":\"
    B827EBFFFF7C1C5F\"},{\" rssi\":-116,\"snr\":- 3.2,\"ts
    \":1486970750881,\"rsig\":[{\"ant\":0,\"chan\":3,\"rssic\":-
     117,\"lsnr\":-3.2},{\"ant\":1,\"chan\":19,\"rssic\":-116,\"
    lsnr\":- 3.2}],\"gweui\":\"7076FFFFFF0103FA\"}],\"data
    \":\"070f2021140a0f5a00\"}" ,"tech":"L"}
```

Senzory komunikující pomocí GPRS

Dále jsou zapojeny dva dataloggery EasyLogGSM od firmy Physicus. EasyLogGSM je univerzální průmyslový datalogger, který disponuje nízko výkonnostním mikrokontrolorem s kvalitním a přesným analogovým převodníkem. EasyLogGSM umožňuje zpracovávat informace okamžitě díky multiúkolovému operačnímu systému, který dokáže flexibilně a spolehlivě zpracovávat probíhající operace.

EasyLogGSM disponuje celkem 12 vstupy. Čtyři analogové unipolární vstupy (AIN1 − 4) disponují 12-bit převodníkem. Další čtyři analogové diferenciální unipolární/bipolární vstupy (A9 − 12) s 24-bit převodníkem. Zbývající čtyři vstupy jsou digitální (DIN1 − 4), a jsou naprogramovány tak, aby byly schopny získávat frekvenci (např. rychlost větru), časový interval (sluneční svit) nebo počet impulzů (použití například u člunkového srážkoměru). Každý z 12 vstupů je definován polynomiálními koeficienty (až třetího řádu, $Výstupní\,hodnota = a_0 + a_1x + a_2x^2 + a_3x^3$, kde x je vstupní hodnota a a_0, a_1, a_2, a_3 jsou koeficienty polynomu) měřených hodnot tak, aby mohly být převedeny na odpovídající veličinu (např. z napětí na teplotu ve stupních Celsia). EasyLogGSM komunikuje pomocí dvou sériových portů RS232/RS485. Jeden port je využit na servisní komunikaci a pro nastavení vstupů. Druhý sériový port pak souží pro komunikaci s GSM/GPRS modemem. GSM/GPRS model je schopen odesílat naměřené výsledky v podobě textového souboru na e-mail či na FTP (*File Transport Protocol*) server. EasyLogGSM disponuje také modulem reálných hodin (*real-time clock − RTC*) napájených z baterie typu CR2032. Čas je synchronizován pomocí GPRS jednou denně. Samotný datalogger je napájen šesti kusy baterií typu AA.

Data jsou odesílána na server firmy Ekotechnika, která zajišťuje servis dataloggeru. Data je možné získat třemi způsoby. Na stránkách `http://envirodata.cz/app/` se po přihlášení zobrazí dostupné dataloggery s možností zobrazení dat a manuálního stažení dat. Dalším způsobem je stáhnutí dat pomocí webového API ve formátu JSON (algoritmus 7). Poslední možností je stáhnout data z SD (*Secure Digital*) karty, kde jsou v jednotlivých souborech (jeden textový soubor za jeden den) uložena data (algoritmus 6). Datalogger měří hodnoty každou hodinu a odesílá data jednou denně. Tyto časové intervaly byly zvoleny v závislosti na velmi energeticky náročném odesílání dat. Alkalické baterie vydrží tento provoz asi jeden týden, lithiové vydrží půl roku. Jelikož datalogger odesílá data jednou denně, je automatické stahování dat pomocí REST API

ideálním řešením. Data z dataloggeru jsou odesílána na speciální email, který spravuje firma Ekotechnika. Data jsou následně ukládána na server, kde je možné k nim přistupovat a stahovat pomocí REST API. Další možností, jak může datalogger odesílat data, je na FTP server. Ovšem využití serverů firmy Ekotechnika, která data zpracuje, zálohuje a publikuje po autorizaci ke stažení pomocí REST API, je z hlediska nákladů a administrace lepší variantou. Více o zpracování dat EasyLogGSM dataloggeru je uvedeno v kapitole 5.3.

V dataloggeru jsou aktuálně připojeny čtyři analogové senzory a jeden digitální. Analogové senzory jsou: Virrib od společnosti Fiedler – elektronika pro ekologii, EC5 od firmy Decagon, HumiAir9 od firmy Ekotechnika (teplota TA a vlhkost RH) a vnitřní teplotní čidlo PT100 ($TAin$). Digitální senzor je srážkoměr SR03 (RG) od společnosti Fielder – elektronika pro ekologii. Data jsou v textovém formátu ukládána následovně.

Algoritmus 6 Ukázka uložených dat v textovém souboru z dataloggeru EasyLogGSM

```
#Name EasyLog035
#SN 035/0213
#(A1) Virrib Avg;
#(A2)    RH Avg;
#(A3)   EC5 Avg;
#(A4)    TA Avg;
#(A8)  Vbatt Avg;
#(A9)   TAin Avg;
#(D1)    RG Avg;
03.11.2016 00:00:00 30.511 78.780 1.017 5.462 9.618 7.747 0.000
03.11.2016 01:00:00 30.499 77.925 1.018 5.767 9.491 7.831 0.000
03.11.2016 02:00:00 30.511 83.539 1.017 4.485 9.594 7.329 0.000
03.11.2016 03:00:00 30.499 80.732 1.018 5.340 9.613 7.554 0.000
```

Algoritmus 7 Odpověď na dotaz na data z Envirodata API

```
{
 "RC":[0,"OK"],
  "table":{
   "data":[
    ["00:00:00 24.12.2019",33.77,86.957,9.77,1.568,5.706,0.0,
     null,7.008],
    ["01:00:00 24.12.2019",33.648,87.079,9.672,1.574,5.706,0.0,
     null,6.907],
    ["02:00:00 24.12.2019",33.599,87.567,9.76,1.579,5.401,0.0,
     null,6.707],
    ...
                ]
        }
}
```

5.2.2 Analytické využití zpracovaných dat

Senzorová data jsou na KGI využívána v řadě environmentálních studií. Mezi hlavní aplikační oblasti patří studium vlivu ekotonů na ekosystémové funkce a zejména při studiu variability půdních vlastností v okolí ekotonů a dále modelování retenční schopnosti a proudění vody v krajině. Obě témata mají mimo jiné společný cíl, podporu precizního hospodaření a šetrného využívání přírodních zdrojů.

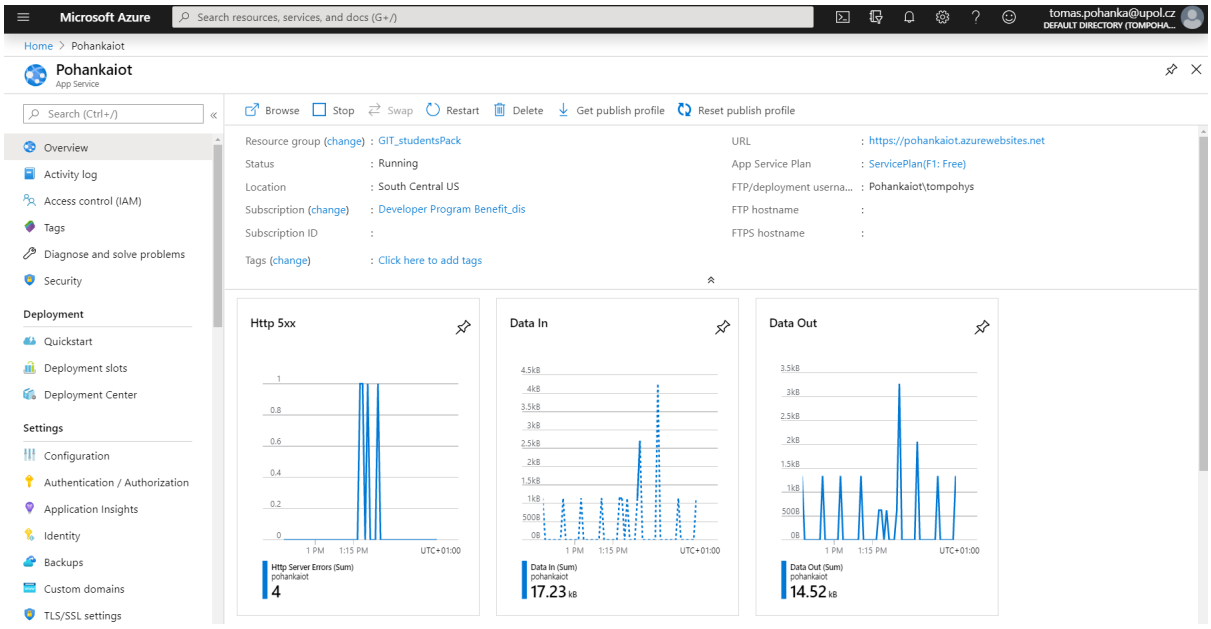
Právě v těchto oblastech je hojně využíváno senzorového měření, které se ukazuje jako vysoce efektivní nástroj pro zajištění dlouhodobého kontinuálního sběru dat v terénu [104].

Realizované studie ([120, 114]) prezentují výsledky studia variability půdního uhlíku. Statistické vyhodnocení potvrzuje, že v blízkosti okraje pozemků kolísají zásoby uhlíku nejen v průběhu roku, ale i v závislosti na vzdálenosti od okraje. Variabilita dostupnosti půdního uhlíku je klíčovým ukazatelem pro agrotechnické opatření.

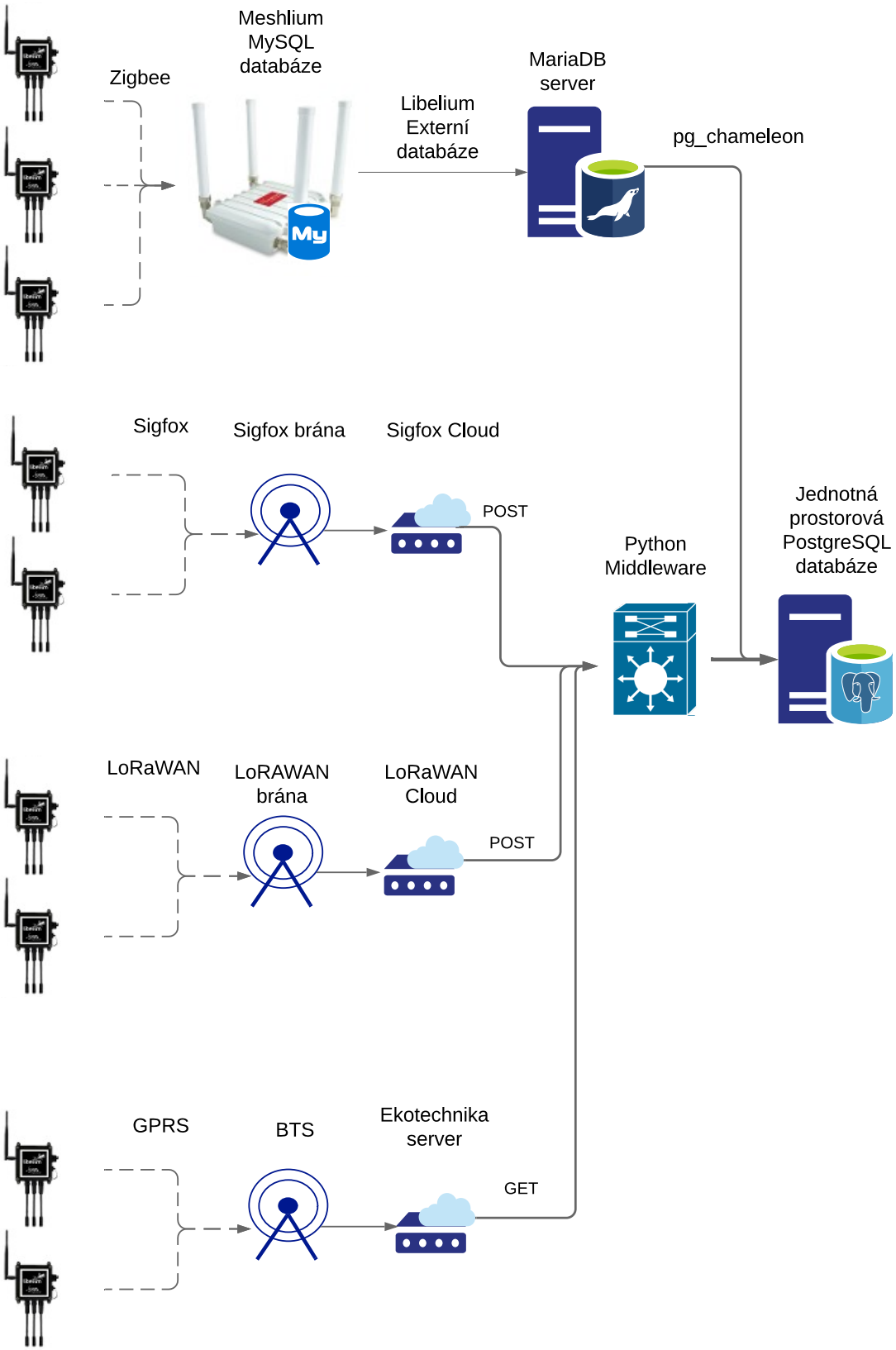
5.3 Integrace senzorových dat do jednotné prostorové databáze

Výsledkem třetího dílčího cíle je funkční middleware, který zajišťuje integraci senzorových dat do jednotné prostorové databáze. Middleware je naprogramovaný v jazyce Python a běží v cloudové službě Microsoft Azure (obrázek 34) na webové adrese *https://pohankaiot.azurewebsites.net* Integrační middleware senzorových dat zapisuje veškerá data do jednotné databáze, vzniká tak centrální místo všech dat. „Zpátečnická“ centralizace (vice v kapitole 4.1.5 na straně 27) je odstraněna pomocí replikačních mechanismů k vytvoření distribuované databázové sítě.

Python middleware využívá web framework Flask, který se stará o příjem dat. Data jsou následně zpracována moduly *psycopg2* a *json*. Tyto knihovny se starají o parsování přijatých dat ve formátu JSON a o odeslání do jednotné prostorové databáze. Modul *psycopg2* se přímo napojí na databázi a odesílá či přijímá data. Čtení dat z databáze slouží pouze k ověření posledního záznamu v tabulce. Python middleware na obrázku 35 je poslední prvek před jednotnou prostorovou databází.



Obrázek 34: Informace o middleware z Microsoft Azure



Obrázek 35: Kompletní schéma přenosu dat ze senzorů do jednotné prostorové PostgreSQL databáze

Jednotná tabulka v databázi slouží k uložení naměřených hodnot jednotlivých čidel senzorů (struktura tabulky je zobrazena v algoritmu 8). Data ze senzorů nelze zpětně opravit. Vše řeší Python middleware, který detekuje a vyhodnocuje nepřesná měření a do jednotné databáze tato data vůbec nezaznamená. Pokud senzor vrací nesmyslná data, například vlhkost vzduchu mimo obor hodnot $0 - 100$ %, nebo teplotu s hodnotou -1000, pak je jasné, že čidlo je buď chybné, nebo má špatný kontakt. Výsledkem ovšem je, že takto chybně poslaná data nelze „opravit“ na správná a dále s nimi není počítáno.

Metadata konkrétních senzorů a čidel jsou poté navázána na statickou část databáze pomocí *sensor* a *id_sensor*, kde *sensor* atribut obsahuje název (či kód) čidla a *id_sensor* obsahuje název (či kód) senzoru. Poloha jednotlivých čidel je ve speciální tabulce *sensor_loc*. Rozsah tabulky *sensor_loc* závisí na povaze senzoru, např. pro zemědělství a krajinu bude senzor delší dobu na jednom místě. Pro sledování vozidel (zejména komunikační technologie GPRS, NB-IoT, LTE-M) bude počet záznamů v tabulce více záviset na frekvenci odesílání nových dat. Struktura tabulky je zobrazena v algoritmu 9. Lze i uvažovat, že změnu polohy může provést i jednotlivé čidlo senzoru, například, pokud se čidlo vymění za kalibrované.

Algoritmus 8 Struktura jednotné tabulky pro senzorová data

```
CREATE TABLE public.data (
    id_sensor text,
    sensor text,
    value numeric(5,2),
    original_id numeric,
    "time" timestamp without time zone NOT NULL,
  id bigint NOT NULL DEFAULT nextval('data_id_seq'::regclass),
CONSTRAINT data_pkey PRIMARY KEY (id, "time") )
```

Algoritmus 9 Struktura tabulky pro záznam polohy

```
CREATE TABLE public.sensor_loc (
    id integer NOT NULL DEFAULT nextval('sensor_loc_id_seq'::
       regclass),
    id_sensor text,
    sensor text,
    time_from timestamp without time zone,
    time_to timestamp without time zone,
    geom geometry,
CONSTRAINT pk_id_sensor_loc PRIMARY KEY (id) )
```

Data do jednotné databáze proudí aktuálně ze tří různých zdrojů a to ze senzorů Libelium se Zigbee a Sigfox (LoRaWAN) technologií a z dataloggerů EasyLogGSM. Z dataloggerů EasyLogGSM jsou odesílána data pomocí GPRS jednou denně, vždy po půlnoci, aby byla daná dostatečná časová rezerva pro zpracování dat firmou Ekotechnika. Middleware je nastaven tak, aby odesílal dotaz na nová data každý den vždy v 7:00. Na straně middleware dochází ke kontrole poslední zapsané hodnoty v databázi (algoritmus 10). Pokud by došlo k jakékoliv chybě, jak na straně middleware či na straně serveru firmy Ekotechnika, budou data stažena za celé období výpadku. Zároveň to řeší i problematiku výskytu duplicitních hodnot.

Algoritmus 10 Získání času poslední hodnoty z tabulky naměřených hodnot

```
cur.execute("SELECT time from data WHERE id_sensor = 'enviro'
   ORDER BY time DESC LIMIT 1;")
last_time = cur.fetchone()
if last_time:
  for last_record in last_time:
    day_after_last_record = last_record + timedelta(days=1)
    day_after_last_record = day_after_last_record.strftime('%d
       .%m.%Y')
```

Data se získávají po přihlášení autorizovaného uživatele přes REST metodou GET, ve které se určí, za jaké období pro který senzor a pro která čidla. Seznam senzorů a odpovídajících čidel lze také získat dotazem na server firmy Ekotechnika (algoritmus

11, první řádek), který opět vrací JSON. Následné vytvoření dotazu na konkrétní data (algoritmus 11, druhý řádek) lze buď manuálně, nebo automaticky parsováním JSON dat z výsledku dotazu na senzory a čidla. Atributy GET metody jsou: a) časové rozmezí; b) označení senzoru a čidla („Loc428-Col1“). Middleware následně každý den získá data ze serveru firmy Ekotechnika v JSON formátu. Vloží naměřenou hodnotu pro každý senzor přímo do jednotné prostorové databáze senzorových dat (algoritmus 12).

Algoritmus 11 Dotaz na data ze senzoru za určité období

```
sensor_list_link = 'http://data.enviroinvest.cz/dta/GetDataJson
    ?list'

link = 'http://data.enviroinvest.cz/dta/GetDataJson?table&
    dateFrom={0}&dateTo={1}&Loc428-Col1&Loc428-Col2&Loc428-Col3&
    Loc428-Col4&Loc428-Col5&Loc428-Col6&Loc428-Col7&Loc428-Col8'
    .format(day_after_last_record, yesterday)
```

Algoritmus 12 Vložení dat do databáze ze serveru Ekotechniky

```
i~= 1
for sens_name in ["Virrib", "RH", "BAT","EC5", "TA", "RG", "
    TAin"]:
  cur.execute("insert into data (time, sensor, value, id_sensor
     ) values ('%s', '%s', %.2f, '%s');" % (timestamp,
     sens_name, data[i], 'enviro'))
  i~+= 1
```

Ze senzorů Libelium jsou data odesílána třemi způsoby, a to pomocí technologií Sigfox, LoRaWAN a Zigbee. Senzory jsou nastaveny tak, aby odesílaly naměřené informace okamžitě po jejich naměření, které je provedeno každých 15 minut. Délka měření závislí na použitých senzorech, např. senzor Sensiron SHT75 má 10 s kalibrační interval, po který je senzor napájen, a po těchto 10 s se teprve měří hodnota teploty a vlhkosti vzduchu. Senzory Libelium disponují RTC modulem, který dokáže zařízení každých 15 minut probudit a vykonat naprogramovanou úlohu. Data jsou přes síť Sigfox odeslána

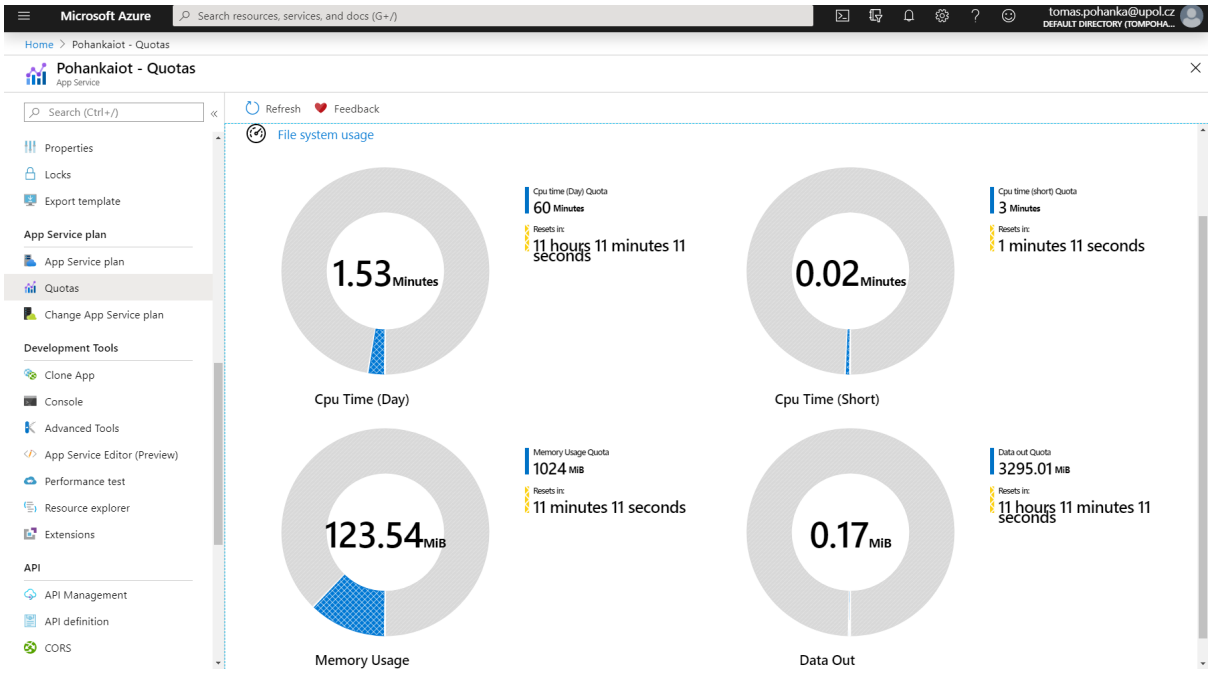
na Sigfox Cloud, odkud jsou pomocí Callback přeposílány na zabezpečený (*SSL*) webový server s middleware. Webový server je nasazen na platformě Microsoft Azure. Ten nabízí zdarma základní prostor pro vytvoření zejména webových stránek. Webový server lze ovšem naprogramovat pomocí Python jazyka, a lze tak využít výpočetní výkon serveru i k příjmu, odeslání a zpracování dat. Služba je omezena časovým využíváním CPU na 60 minut za den a 1 GB RAM (*Random Access Memory*). Aktuálně jsou na server posílány dotazy od dvou senzorů přes Sigfox Cloud a je využito asi tři minuty CPU času a 70 MB RAM (obrázek 36 na straně 107). Nebyl by tedy problém obsluhovat na zdarma poskytované platformě i 30 senzorů, i když v omezené míře. Middleware, který běží na Microsoft Azure, lze takto provozovat na jakémkoli vlastním serveru, čistě s podporou Python jazyka. Hlavní část middleware, který se stará o příjem, zpracování a uložení přijatých dat ze Sigfox Cloud, je zobrazen v algoritmu 13. Middleware využívá webový framework Flask, který je spolu s Django jedním z nejpoužívanějších [11].

Algoritmus 13 Část Python middleware pro příjem, zpracování a odeslání dat

```
@app.route('/', methods = ['POST'])
def home():
 if request.headers['Content-Type'] == 'application/json':
  if request.headers['auth'] == 'disertacniprace2020':
   with psycopg2.connect("dbname='sensor' user='**********' host
      =158.194.94.*** password='**********' ") as con:
    cur = con.cursor()
    cur.execute("INSERT INTO public.sigfox (value) VALUES ('%s')" %
       (json.dumps(request.json)),)
    myjson = request.json
    time = myjson['time']
    device = myjson['device']
    rest_s = 0
    rest_t = 0
    for sensor in myjson:
     if sensor == 'rest':
      rest_s = str(myjson[sensor])[0]
      rest_t = str(myjson[sensor])[1]
      for sensor in myjson:
       if sensor not in ['time', 'device', 'data', 'rest']:
        # v~sensoru přidávám 50 aby -50 stupnu byla 0 a~50 stupnů
           byla 100
        value = myjson[sensor]
        if sensor == 'TCA':
            if value > 0 and value] < 100:
              value = float(value) - 50 + float(rest_t)*0.1
        elif sensor == 'TCB':
          if value > 0 and value < 100:
            value = float(value) - 50 + float(rest_s)*0.1
      cur.execute("INSERT INTO public.data (id_sensor, sensor,
         value, time) VALUES ('%s','%s', %.2f, '%s')" % (device,
         sensor, float(value), datetime.fromtimestamp(int(time)).
         strftime('%d-%m-%Y %H:%M:%S')),)
return render_template('index.html', title='Home Page', year=
   datetime.now().year, data = "sended")
```

Dekorátor @app.route('/', methods = ['POST']) určuje, že přímo na doménovém jménu nejvyššího řádu (například .cz, .com) může přijímat data pouze metodou POST. Vzápětí následuje první podmínka na data, zda mají v hlavičce uvedeno, že se jedná o JSON data. Následuje připojení k databázovému serveru, který běží na KGI.

Jako absolutní záloha, jsou data takto přijatá uložena do tabulky *sigfox* v databázi *sensor*. Tato tabulka obsahuje pouze identifikátor (datový typ *serial*) a data (datový typ *jsonb*). PostgreSQL umí pracovat s dokumenty JSON a XML přímo. Proto lze vytvářet dotazy i přímo na tento JSON dokument. JSON ovšem slouží jako záloha primárních nezpracovaných dat a dotazy na JSON objekt jsou proto mírně pomalejší, než dotazy na klasickou tabulku. Aby bylo zamezeno tzv. SQL Injection, tedy podvrhnutí škodlivého kódu, Sigfox Cloud umožnuje do šifrované hlavičky zadat i autentizační uživatelské údaje. Kód dále rozloží JSON dokument na jednotlivé elementy, které zpracuje a uloží do jednotné tabulky *data* pro všechny senzory.



Obrázek 36: Dostupné zdroje middleware na Microsoft Azure

Dalším integračním prvkem, který je nutné řešit je zpracování dat ze senzorů Libelium, jež komunikují pomocí technologie Zigbee. Základní popis technologie je uveden v kapitole 4.3.1.4. Senzory komunikující na technologii Zigbee komunikují s bránou. U firmy Libelium se brána jmenuje Meshlium a je postavena na operačním systému Debian a databázi MySQL 5.0.51a. Senzorová data jsou ukládána do tabulky

sensorParser a její struktura je popsána v algoritmu 14. Nejdůležitější pole jsou *id* s automaticky navyšovaným identifikačním číslem, *id_wasp* s označením konkrétního senzoru, *sensor* s označením konkrétního čidla, *value* s naměřenou hodnotou (v textovém formátu), *timesptamp* s časovou značkou přijmutí dat a nakonec *sync*, které označuje, zda byla data odeslána na externí server.

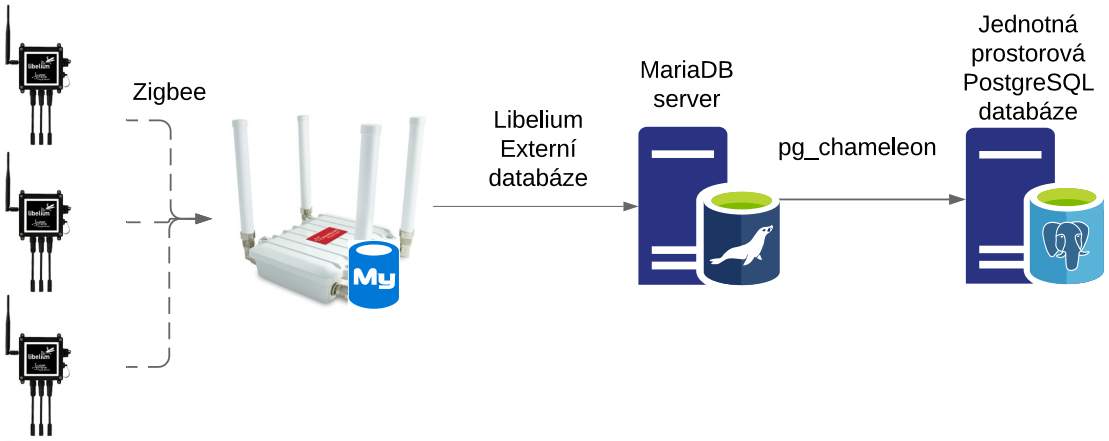
Algoritmus 14 Struktura tabulky pro senzorová data uložená v bráně Meshlium

```
CREATE TABLE IF NOT EXISTS `sensorParser` (
  `id` int(11) NOT NULL auto_increment,
  `id_wasp` varchar(16) NOT NULL,
  `id_secret` varchar(10) NOT NULL,
  `frame_type` int(11) default NULL,
  `frame_number` int(11) default NULL,
  `sensor` varchar(16) NOT NULL,
  `value` varchar(65) NOT NULL,
  `timestamp` timestamp NOT NULL default CURRENT_TIMESTAMP,
  `sync` bigint(12) unsigned NOT NULL default '0',
  `raw` varchar(100) NOT NULL default 'noraw',
  `parser_type` tinyint(3) NOT NULL default '0',
  PRIMARY KEY (`id`), KEY `time` (`timestamp`) )
ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=2997481 ;
```

Odeslání dat na externí server je velmi vítaná funkce přímo implementovaná v bráně Meshlium. Externí databáze ovšem může být pouze MySQL nebo MariaDB, a tabulka musí mít shodnou strukturu. Hlavní databáze v Meshlium pak jen přiřazuje příznak v atributu *sync*, zda byl záznam úspěšně odeslán na externí server. Jako externí server byl použit jeden z počítačů Katedry geoinformatiky. Na serveru je nasazen balíček XAMPP (*Cross-Platform, Apache, MariaDB, PHP a Perl*) ve verzi 5.6.28, který instaluje a propojuje Apache HTTP Server, MariaDB a PHP. Na tento server jsou následně odesílána data z Meshlium. Důvod vzniku tohoto serveru a přeposílání dat má několik důvodů: a) záloha primárních dat; b) rozložení zátěže; c) bezpečnost z hlediska přístupů do databáze.

Záloha primárních nezpracovaných dat by měla být součástí každého projektu. Nejhorší situace nastane, pokud celý senzor zkolabuje (nejčastěji vybitá baterie). Chybu čidla lze vypozorovat z přijatých dat. Pokud není dostupná brána pro příjem dat, lze data zpětně načíst z vnitřní paměti senzoru. Při nedostupnosti externího serveru budou data automaticky přeposlána z Meshlium brány (podle atributu *sync*). Každá havárie se dá ošetřit upozorněním na nečekanou událost, například v databázovém triggeru nebo middleware.

Architektura senzorů komunikujících na technologii Zigbee od firmy Libelium synchronizuje data do jednotné databáze bez účasti naprogramovaného integračního middleware. Tím, že jednotná prostorová databáze senzorových dat využívá PostgreSQL a externí databáze Meshlium brány je MariaDB, je využit nástroj *pg_ chameleon*. Ten vytváří replikační spojení MariaDB a PostgreSQL. Replika je nastavena tak, že PostgreSQL je *slave* server MariaDB serveru. Jedná se o asynchronní logickou replikaci. Výkon nástroje a celkové zdržení při replikaci je u asi 2 000 nově vložených záznamů za sekundu do MariaDB znamenalo asi 10 s zpoždění v PostgreSQL [45]. Schéma toku dat je zobrazeno na obrázku 37. Pg_chameleon dokáže replikovat jednotlivé tabulky z databáze. Nelze ovšem definovat konkrétní atributy tabulky. Je tedy replikována celá tabulka. Tím, že jednotná prostorová senzorová databáze nevychází přímo z Libelium Meshlium databáze, jsou data ze senzorů uložena nejdříve do samostatné tabulky, a následně pomocí funkce vkládána do jednotné tabulky. Tato modifikace struktury může probíhat na MariaDB serveru, tak na PostgreSQL. Funkce při každém novém přidaném záznamu do tabulky z MariaDB (pomocí spouště (*trigger*)) ověří minimálně, že data jsou v zadaném rozsahu hodnot, a poté je uloží do jednotné tabulky. Algoritmus 15 popisuje vkládání dat z replikované tabulky z MariaDB do jednotné tabulky *data*. Za touto funkcí následuje vytvoření spouště nad tabulkou z MariaDB, tak, aby se funkce při každém novém záznamu spustila.



Obrázek 37: Schéma propojení senzorů se Zigbee komunikační technologií s jednotnou PostgreSQL databází

Algoritmus 15 Funkce a spoušť pro úpravu struktury původní Libelium Meshlium tabulky.

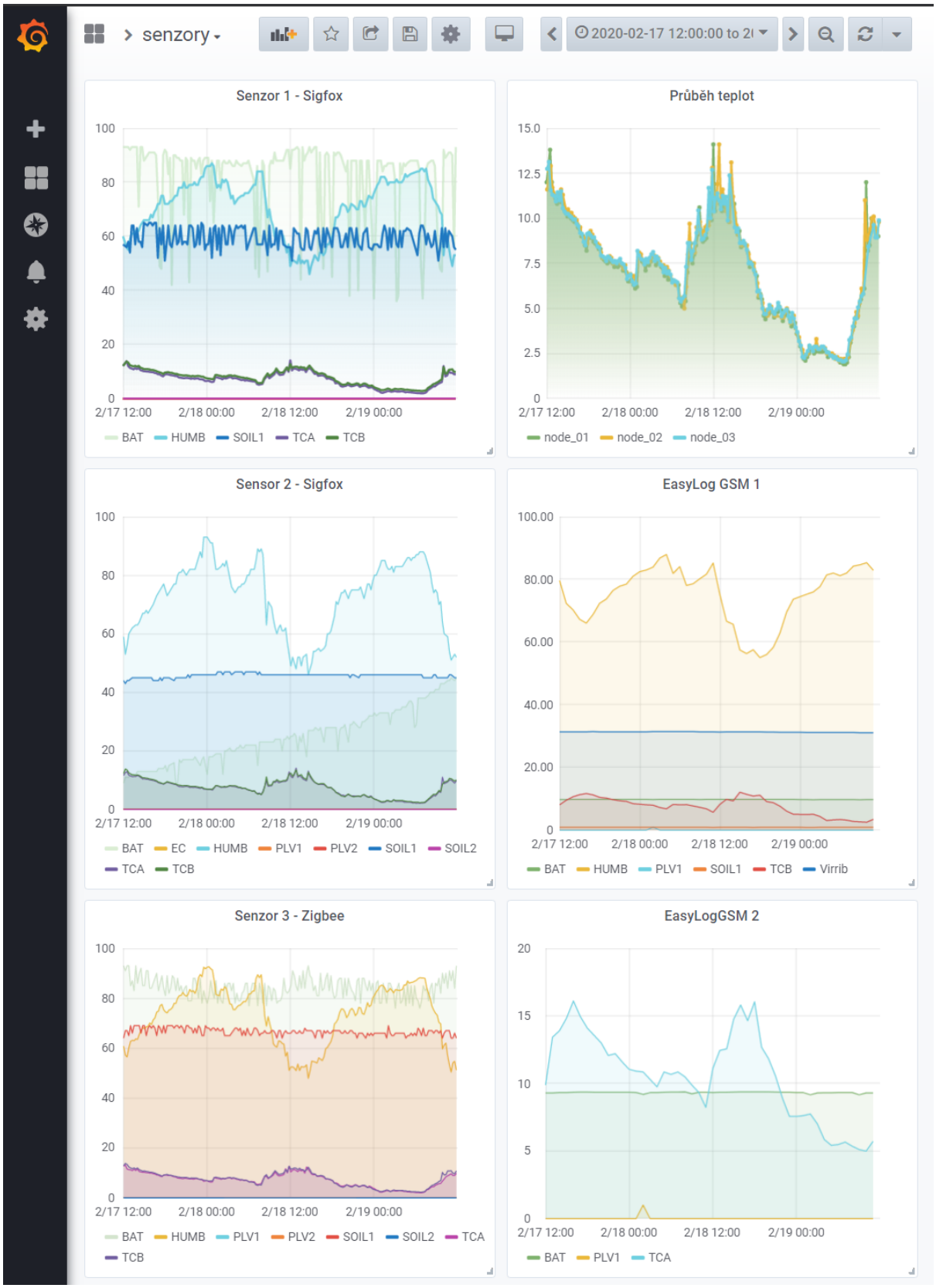
```
CREATE OR REPLACE FUNCTION public.libelium_data()
RETURNS trigger AS $BODY$
BEGIN
   IF to_number(NEW.value, '99999.99') > -999 OR to_number(NEW.
      value, '99999.99') < 999
  THEN INSERT INTO data(id_sensor, sensor, value, time,
      original_id)
  VALUES (NEW.id_wasp, NEW.sensor, to_number(NEW.value, '999.99
      '), NEW.timestamp, NEW.id);
  END IF;
RETURN NEW;
END $BODY$
LANGUAGE plpgsql
VOLATILE    COST 100;

CREATE TRIGGER libelium_data
AFTER INSERT ON sammwap.sammwap
FOR EACH ROW
EXECUTE PROCEDURE public.libelium_data();
```

S rostoucím počtem senzorů a délkou sběru dat vyvstává otázka ohledně dělení tabulky (*table partitioning*). Obecným předpokladem pro vytváření částí tabulky je, že velikost tabulky by neměla přesáhnout velikost RAM. Jediná tabulka může mít například v PostgreSQL až 128 TB (kapitola 4.1.2.1 na straně 21) a neomezený počet řádků. I hodnota 128 TB dat v jedné tabulce může být za určitý čas dosažena. Aby bylo možné tento limit obejít, lze vytvářet části tabulky, kde bude mít každá část tabulky určité velikostní omezení. Ať již se jedná o velikost počtu sloupců (vertikální členění), typ záznamů (historické a aktuální záznamy; horizontální členění), nebo statický (např. 1 000 000 řádků) či dynamický (týdenní části) počet záznamů či využití vertikálního, horizontálního a dynamického členění zároveň. Tohoto mixu využívá například rozšíření TimescaleDB pro PostgreSQL [86]. Části tabulky se ovšem zobrazují a chovají jako by byla tabulka pouze jedna. O dynamické slučování pro zobrazování se stará interní systém databázového serveru.

Dělení tabulky by se mělo opírat i o reálné využití dat v praxi. Zejména dotazy na data budou podstatně rychlejší, pokud bude většina dotazů z jedné části. Například, pokud velmi často směřují dotazy na data za jeden měsíc, je vhodné nastavit dělení tabulky po měsících tak, aby byly dotazy směrovány do jednoho celku. I dotazy na týdenní části tabulky budou řádově rychlejší než dotaz na nedělenou tabulku se stovkami milionů záznamů a velikostí několik desítek GB. Mimo zvýšení propustnosti čtení dat je zvýšen i samotný zápis dat. Data se připisují jen k části dat, a ta v určitý moment vytvoří další část. Nevýhodou je samozřejmě vyšší režie na administraci databáze. Další nevýhodou je při dělení tabulky nutné využít takový typ replikace, u které je možné replikovat i změnu schématu.

Pro vizuální představu lze data jednoduše zobrazit do grafu. Lze například využít webovou aplikaci Grafana, která nabízí přímé napojení na PostgreSQL. Tím lze veškerá data přehledně zobrazit v grafu (obrázek 38). Grafana dovoluje nastavit krok aktualizace,

ale jedná se o klasický model klient – server. Když do databáze budou přidána nová data, Grafana tato data nezobrazí dokud uživatel nepošle požadavek na aktualizaci. Grafana data pouze zobrazuje, žádné analýzy zde provést nelze. Dalším možným zobrazením dat i s využitím prostorových informací je řešen v autorově studii [121].



Obrázek 38: Zobrazení naměřených dat pomocí senzorů v grafech.

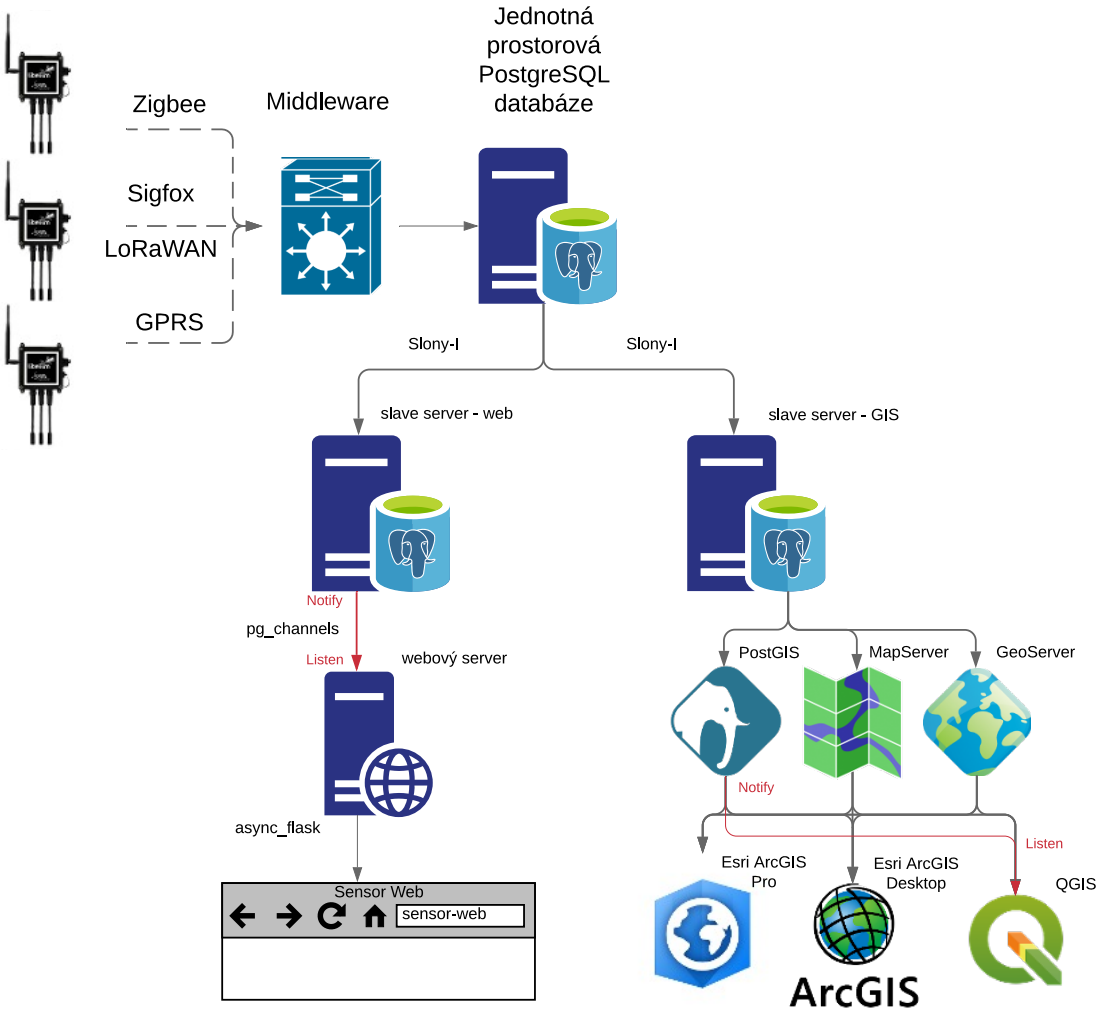
5.3.1 Distribuování jednotné prostorové databáze senzorových dat

Integrace všech senzorových dat do jednotné prostorové databáze vedlo k její centralizaci. Pro zachování vysoké dostupnosti, optimální výkonosti a minimální zálohu dat, je vhodné centralizovanou databázi distribuovat. O distribuovaných databázích je více pojednáno v kapitole 4.1.5 na straně 27. Pro PostgreSQL existuje mnoho možností, jak distribuovat databázi. Dva prakticky realizované přístupy, z teoreticky několika možných řešení, jsou popsány níže.

Varianta 1 – s využitím rozšíření Slony-I

Hlavním záměrem bylo vytvoření funkční cesty ze senzorů až po QGIS využívající PostgreSQL funkce *Notify/Listen*.

První vytvořený přístup využívá Slony-I replikačního systému pro vytvoření replikačního klastru. Jednotná prostorová databáze, která je plněna senzory komunikující na různých technologiích a která vychází z výsledků dílčích cílů je ve schématu 39 využívána jako *master* databáze. Z této databáze jsou následně vytvořeny dvě asynchronní repliky. Dvě repliky proto, aby mohli být odděleni weboví a desktop uživatelé, a tím se ještě více rozložila zátěž. Pro webového uživatele nemusí být databázový server tak výkonný. Naopak, pro desktop GIS klienty je vhodnější použít výkonný server. Rozlišení databáze pro web i pro GIS je na základě přidělené IP adresy serveru. Pro web a GIS *slave* servery je rozhodující jejich schopnost číst a odesílat data. Tento přístup je vhodný, pokud se počet záznamů pohybuje v desítkách milionů až nižších jednotek stovek milionů v závislosti na velikosti volné RAM serveru. V navržené struktuře jednotné prostorové databáze představuje 1 000 000 záznamů asi 150 MB. U serveru s 16 GB RAM je u počtu 100 000 000 záznamů, tedy asi 15 GB dat v jedné tabulce hranicí, kdy by se podle doporučení měla vytvořit první část tabulky při horizontální dělení. 100 000 000 záznamů se při počtu 1 000 senzorů, které budou posílat data každých 30 minut dosáhne za 5,7 let.



Obrázek 39: První přístup distribuovaného prostředí

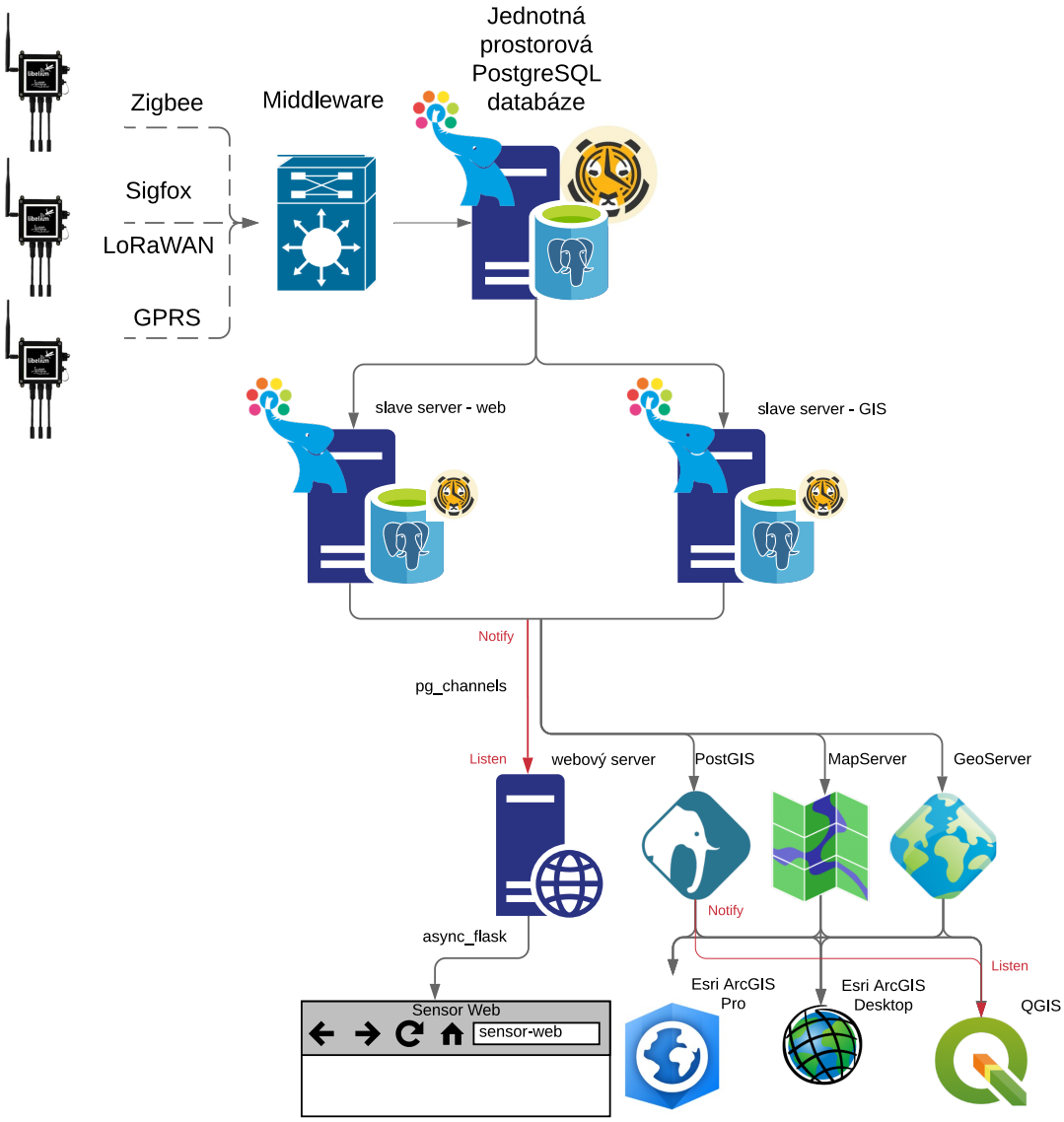
Varianta 2 – s využitím rozšíření PgPool a TimescaleDB

U druhého přístupu byla provedena inicializace PgPool, bylo přidáno rozšíření TimescaleDB a vyzkoušena replikace mezi *master* a *slave* databází. Druhý návrh vychází ze získaných zkušeností a dokumentace k PgPool a TimescaleDB.

Tento přístup využívá databázové dělení do částí pomocí TimescaleDB i PostgreSQL streaming replikaci a middleware pro vyrovnání zátěže a celkový management replikačního klastru PgPool-II. Při této konfiguraci již téměř nezáleží na počtu senzorů posíla-

jící data. TimescaleDB vytváří části tabulky přesně podle požadavků s přihlédnutím na způsob využití dat (čtení poslední hodnoty, čtení hodnot z posledního týdne, měsíce) i s přihlédnutím na hardware serveru, zejména pak na velikost RAM.

Pokud budou uvažovány stejné hodnoty jako u prvního přístupu, pak pro jeden oddíl na jeden měsíc při uvažované velikosti 30 GB (asi 200 000 000 záznamů) a při odesílání dat senzorem každých 10 minut (nejvyšší počet zpráv za den pro LPWAN) by bylo potřeba 46 300 senzorů. TimescaleDB by za jeden měsíc vytvořil další horizontální oddíl, a díky neomezené velikosti PostgreSQL databáze lze tyto oddíly vytvářet neustále. PgPool běží na každém serveru z důvodu pokrytí výpadku jedné z databáze. Při výpadku *master* databáze se okamžitě povýší jeden ze *slave* serverů a nedojde tak k žádnému výpadku. Jakmile bude původní *master* databáze opět funkční, synchronizují se provedené změny z aktuální *master* databáze a z původní *master* databáze se stane *slave* databáze. Klient (Python middleware, QGIS, GeoServer ...) tuto výměnu vůbec nepocítí, jelikož s databázovým klastrem komunikuje pomocí virtuální IP adresy pro celý klastr. PgPool pak sám rozhodne, na kterou z databází klienta připojí. Pokud přijde požadavek na zápis, přepošle data na *master* databázi (jednotná prostorová PostgreSQL databáze či právě aktuální *master* databáze), pokud přijde dotaz na čtení, využije se jeden ze *slave* serverů. Jakýkoli uživatel, ať již s požadavkem na zápis či čtení, se připojuje pouze k jedné IP adrese.



Obrázek 40: Druhý přístup distribuovaného prostředí

5.3.2 Aplikační využití

Pro vytvoření funkční cesty od senzorů až po QGIS byla využita funkcionalita *Notify/Listen*. Poprvé byla představena v QGIS 3 (vydána 24. 2. 2018). Jedná se o nativní funkcionalitu PostgreSQL od verze 9 (vydána 20. 9. 2010), která není závislá na žádném dalším modulu.

V PostgreSQL databázi je nutné vytvořit funkci (algoritmus 16) a spouštěč (algoritmus 17), která bude zachytávat změny v tabulce *data* a následně funkcí odesílat zprávu o nových, změněných či smazaných záznamech. Tím je na straně databázového serveru vytvořen (publikován) kanál pro odběr (*subscription*) zpráv. Kdykoliv se v tabulce *data* jakkoli změní data, odešle se zprávě kanálem *qgis*.

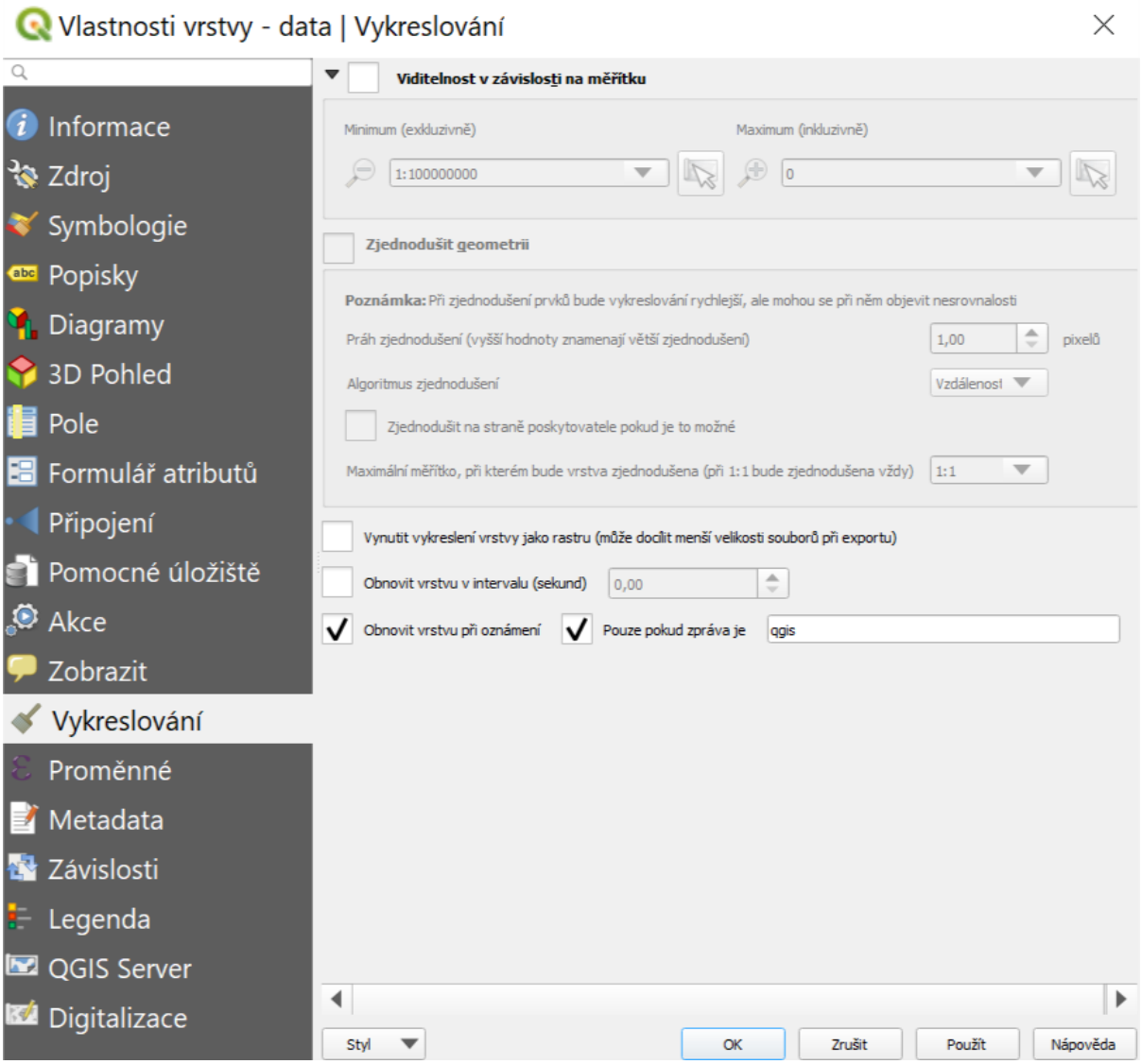
Algoritmus 16 Funkce pro odesílání notifikací z PostgreSQL do QGIS

```
CREATE FUNCTION public.fce_notify_qgis()
  RETURNS trigger
  LANGUAGE 'plpgsql'
  COST 100
  VOLATILE NOT LEAKPROOF
AS $BODY$
    BEGIN NOTIFY qgis, 'qgis';
    RETURN NULL;
    END;
  $BODY$;
```

Algoritmus 17 Spoušť (trigger) pro zavolání funkce fce_notify_qgis při zvolené události

```
CREATE TRIGGER trig_qgis_notify
AFTER INSERT OR DELETE OR TRUNCATE OR UPDATE
ON public.data
FOR EACH STATEMENT
EXECUTE PROCEDURE public.fce_notify_qgis();
```

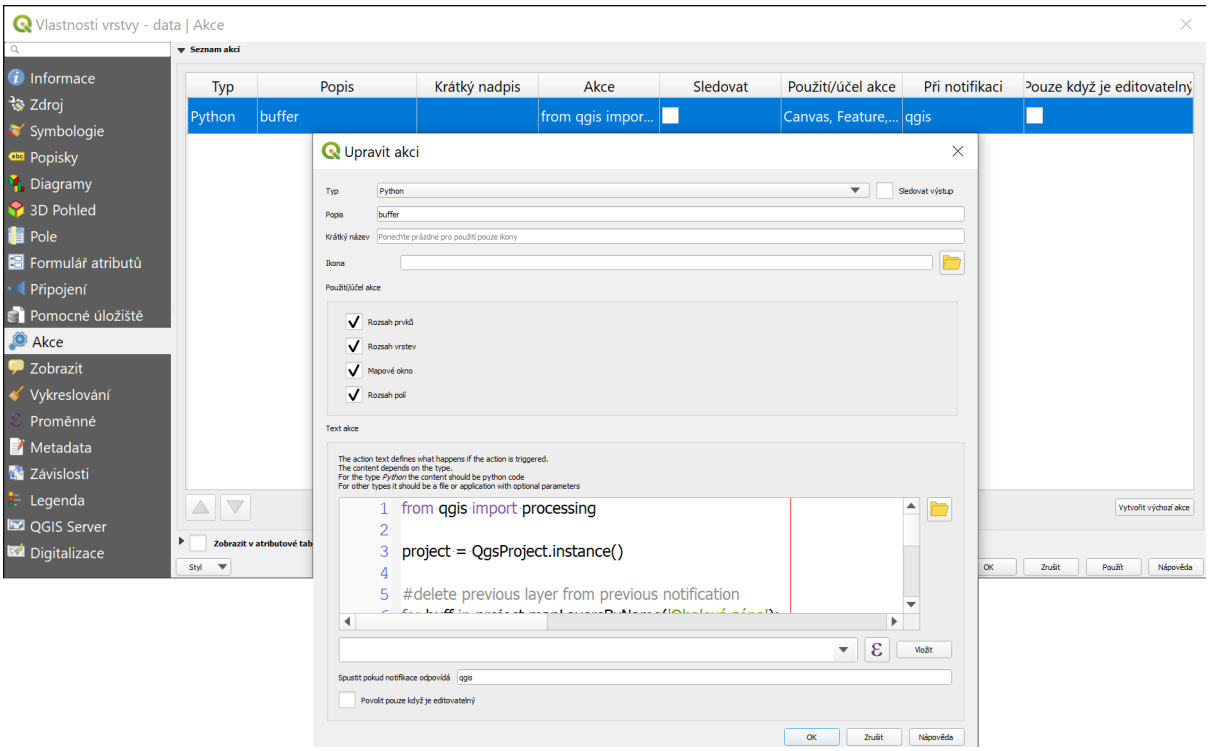
Na straně QGIS je nutné se připojit k PostGIS databázi. U prvního přístupu distribuovaného prostředí (obrázek 39 na straně 115) je to databáze *Slave server – GIS*, u druhého přístupu (obrázek 40 na předchozí straně) se QGIS připojí k virtuální IP adrese. Po přidání požadované vrstvy do QGIS, se u těchto dat nastaví odběr kanálu *qgis* (obrázek 41).



Obrázek 41: Nastavení odběru u vrstvy v QGIS

Nastavení odběru zajišťuje, že při změně dat v jednotné prostorové databázi senzorových dat se projeví změna i v QGIS bez uživatelské interakce. Díky možnosti nastavení a spuštění vlastní *akce* na základě notifikace lze takto vytvářet plně automatické analýzy. Na obrázku 42 je vidět samotné nastavení tohoto makra a v algoritmu 18 poté samotný kód, zajišťující automatickou analýzu. Makra lze vytvářet pro obecný systém, ale i pro Windows, MacOS, Linux či čistě jen otevřít soubor (například obrázek). Makro ovšem lze především naprogramovat v Python, takže jakákoliv úloha, která lze vytvořit v QGIS

může být zde naprogramována. Vytvořený ukázkový algoritmus (18) vytváří Obalovou zónu (*buffer*) okolo bodů. Algoritmus by samozřejmě mohl vytvářet i interpolace a další pokročilé analytické úlohy. Zde se jedná pouze o ukázku možného dalšího využití tohoto nástroje.



Obrázek 42: Nastavení makra v QGIS

Algoritmus 18 Algoritmus makra pro vytváření obalové zóny

```
from qgis import processing
project = QgsProject.instance()

# vymaž starou vrstvu z~předešlé notifikace
for buff in project.mapLayersByName('Obalová zóna'):
  project.removeMapLayer(buff.id())

#vytvoř Obalovou zónu
result = processing.runAndLoadResults("native:buffer",
            {'INPUT': 'data',
              'DISTANCE': 0.1,
              'SEGMENTS': 10,
              'DISSOLVE': True,
              'END_CAP_STYLE': 0,
              'JOIN_STYLE': 0,
              'MITER_LIMIT': 10,
              'OUTPUT': 'memory:test'})
result_layer= result["OUTPUT"]

#nechť je vrstva viditelná
project.layerTreeRoot().findLayer(result_layer).
   setItemVisibilityChecked(True)
```

6 DISKUZE

V průběhu tvorby práce docházelo k řadě rozhodnutí, kterou cestou se výzkum bude ubírat. Bylo nutné se rozhodnout nad výběrem technologií, komponent, testovaných algoritmech apod. Z toho důvodu lze předložený výzkum a navržené řešení označit jako autorský přístup, který není jediný možný, ale je zvolen s nejlepším vědomím a svědomím a na základě dostupných znalostí a prostředků. Dosažené výsledky a ověřený provoz jsou však důkazem, že výzkum se ubíral správným směrem.

Výsledky prvního dílčího cíle byly zaměřeny na replikační možnosti databázových systémů PostgreSQL a MySQL. Výběr databázových systémů (tabulka 1 na straně 7) podléhal zvoleným kritériím, kde velkou váhu má využití databázových systému na KGI, ale i snadné rozšíření odborné komunitě pracující s nízkonákladovými senzory a nízkonákladovou, nízkoenergetickou komunikační technologií. Výběr replikačních mechanismů pro PostgreSQL a MySQL vycházel z rešeršní části a i z prací vytvořených na KGI (práce Solanská [141] a Trnová [146]). U MySQL lze využít jen vestavěné funkce, kdežto u PostgreSQL lze využít mnoho existujících rozšíření, kdy každé z nich poskytuje rozdílné metody replikace. Slony i PgPool se velmi často objevoval v odborných publikacích a tématických diskusích a neustálý vývoj nástrojů zaručuje jejich podporu i v nejnovějších verzích PostgreSQL [138, 24].

Dalším významným rozhodnutím bylo omezení přenosové rychlosti na 10 Mbps. To bylo zvoleno s ohledem na dnešní technologie, například IEEE 802.11b z roku 1999 komunikovalo 11 Mbps či HSPA+ (*High Speed Packet Access*), někdy označováno jako 3,5G komunikovalo právě rychlostí 10 Mbps. Dnes již lze komunikovat rychlostmi až 10 Gbps (*Gigabit per second*). Tyto rychlosti jsou předpokládány u páté generace mobilních sítí (5G), stávající WiFi6 (ve dřívějším označení 802.11ax) má teoretický limit při 9,6 Gbps a budoucí WiFi7 (802.11be) až 30 Gbps. UTP patch kabel kategorie 5 má limit 100 Mbps, 5E má 1 Gbps, kategorie 6 a 7 má maximum 10 Gbps a připravovaná kategorie

8 má maximum při 40 Gbps [22]. Rychlost 10 Mbps je z dnešního pohledu tedy extrémně nízká rychlost, který byla zvolena jen pro tento test. Omezení rychlosti má simulovat reálný provoz serveru, kdy na jednom zařízení běží více softwarových serverů či služeb. Typicky na jediném hardwarovém serveru jsou souběžně spuštěny webový, databázový i aplikační server, které využívají sdíleného Internetového připojení. V rámci předtestování bylo také zjištěno, že velký vliv na celou replikaci má centrální procesorová jednotka (*CPU*) [54]. Proto byla měřena nejen vytíženost síťového připojení ale i vytíženost CPU. Pro měření a omezení rychlosti připojení byl využit program NetLimiter 4, pro měření vytíženosti CPU byl využit HWMonitor Pro v1.3.

Hodnocení replikačního klastru se opíralo o stávající metody hodnocení výkonu databázového serveru jako jsou TPC-x (*Trasaction Processing Performance Council*), SSB (*Star Schema Benchmark*) či YCSB (*Yahoo! Cloud Services Benchmark*) [1, 40, 81, 54]. Tyto hodnotící testy databáze zjišťují například maximální počet uložení za sekundu, čas odezvy při určitém počtu uživatelů či čtecí a editační testy. Ty se ovšem zaměřují převážně na číselné a textové datové typy.

Výhody distribuovaných (prostorových) databází se v malém měřítku téměř neprojeví a spíše převáží nevýhody ve formě vyšší náročnosti na spuštění a celkovou administraci distribuované sítě. Ale už při jednotkách souběžných připojení na jedinou databázi může být server ovlivněn minimálně zvýšenou odezvu [7]. Replikační databázové sítě (klastry) jsou nejvíce vidět u mezinárodních firem, kde mezikontinentální zpoždění v komunikaci je již značné. Geo-replikace (replikace s cílem snížit komunikační zpoždění na velkých vzdálenostech) jak jí označuje Microsoft Azure či Cross-Region replikace u Amazon S3, využívají zejména firmy, které nabízejí cloudové služby. Pro služby typu Microsoft Azure, Amazon AWS, Maptiler Cloud, se již bez geo-replikace neobejdou.

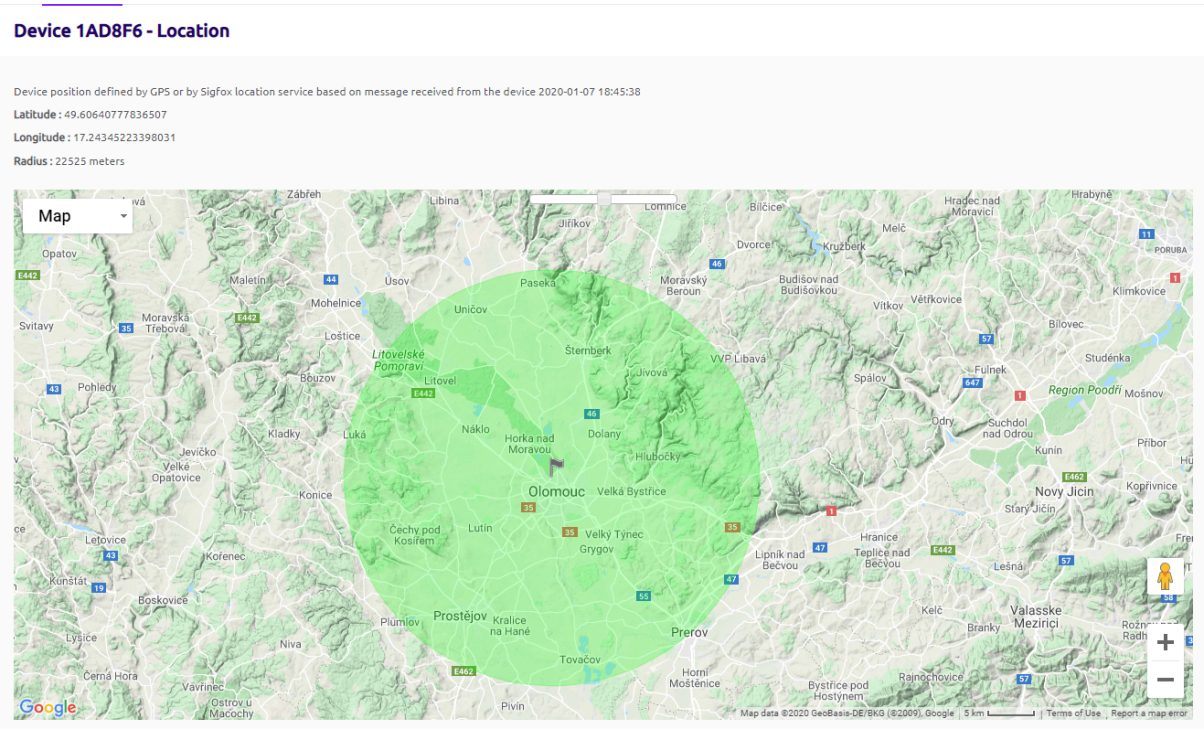
Výsledky druhého dílčího cíle vyřešily problematiku heterogenních senzorových sítí s cílem umožnit přenos a zpracování dat, které usnadní jejich následnou integraci dat do jednotné prostorové databáze. Senzory byly úmyslně osazeny rozdílnými komunikačními technologiemi, aby se zvýšila heterogenita testovaných sítí a tím se zvýšila potřeba naprogramování vlastního integračního middleware (výsledek třetího dílčího cíle). Komunikační technologie jsou z oblasti LPWAN (až na GPRS), které umožňují komunikaci na velkou vzdálenost s velmi malou energetickou náročností. LPWAN technologie mají i své nevýhody, jako například omezené pokrytí signálem, v některých případech nízké časové rozlišení či odesílání dat v objemech jednotek bytů [102]. Představují však možné řešení problému s využitím senzorových měření i v místech, kde není zajištěn pevný zdroj energie, například v lesích, polích či horách.

Míra komprimace přenášených dat byla před odesláním ze senzorů upravena tak, aby každé čidlo odesílalo celočíselnou hodnotu v rozsahu $0 - 255$ (1 B v hexadecimálním zápisu 0 − FF). Je to z důvodu využití parsování dat na straně Sigfox Cloud a odeslat JSON již s identifikací konkrétního čidla spolu s naměřenými hodnotami. V samotném senzoru mohla být naprogramována vyšší komprimace dat například pomocí LZW (Lempel-Ziv-Welch kompresní algoritmus) či RLE (*Run-length encoding*). Data by se pak místo na cloudu služby Sigfox či LoRaWAN parsovala až v samotném middleware, kde by se hexadecimální řetězec dekomprimoval a až následně by se hodnoty přiřadily konkrétním čidlům.

Odesílání dat je možné řešit několika způsoby a byl vybrán ten, který je možné využít jak pro službu Sigfox tak pro LoRaWAN. Bylo zvoleno odesílání dat ve formátu JSON pomocí metody POST na SSL zabezpečený server. U Sigfox Cloud je možné si vytvořit vlastní strukturu JSON, kdežto z LoRaWAN IoT Portálu lze data odesílat pouze v nadefinovaném formátu [12]. Pro příjem zpráv z uživatelského zpětného volání (*Callback)* lze vybrat ze dvou možností, a to jako *jednoduchá*, nebo *pokročilá* data.

Pokročilá data na rozdíl od *jednoduchých* podporují: a) výpočet polohy senzoru na základě síly přijatého signálu bránami; b) fixní polohu zadanou u metadat senzoru; c) kód země ve které se senzor nachází. Vypočítaná poloha má podle síly přijatého signálu přesnost řádově jednotky km v závislosti na počtu bran, které zprávu přijaly. *Pokročilá* data mohou být až o 30 s zpožděná oproti *jednoduchým* datům.

Fixní poloha senzoru může být změněna přímo na administrační stránce v Sigfox Cloud nebo pomocí REST API přes metodu PUT. Využití vypočtené polohy je tedy velmi omezené a lze ji využít všude tam, kde přesnost i několik jednotek kilometrů nehraje roli, např. pro hrubé sledování zásilky, průběžné sledování mezinárodní dopravy (obrázek 43), apod.



Obrázek 43: Odhad polohy senzoru na základě výpočtu Sigfox Cloud

Pokud uživatel zná polohu senzoru, je možné ji přímo zadat do Sigfox Cloud. Změny polohy lze následně provést pomocí REST API nebo ručně na webové stránce vložením přímo ke každému senzoru, a tím využívat příjem pokročilých dat ze Sigfox

Cloud i se zadanou polohou. Pokud uživatel disponuje informací o aktuální poloze, lze ji i přímo zapsat do databáze, a tím zkrátit zpoždění zprávy, snížit datový tok a nepřijímat duplicitní informace o poloze.

Sigfox Cloud nabízí i přednastavou funkci zpětného volání *(Callback),* která existuje pro AWS IoT (*Amazon Web Services*) a Kinesis. Dále pak pro Microsoft Azure IoT Hub a Event Hub a i pro IBM Watson IoT Platform [25]. Zmíněné PaaS cloudové služby mohou přijímat miliony zpráv za sekundu pomocí HTML, MQTT, AMQT atp. protokolů, jsou škálovatelné, bez složitých konfigurací a bez administrace vlastního serveru. Výkon je ovšem za poplatek od jednotek korun za hodinu až po několik tisíc.

Při zpracovávání pokročilých časoprostorových analýz se často využívají data z různých zdrojů s různým měřítkem či prostorovým rozlišením. U strojových senzorových dat lze integraci vyřešit i právě pomocí middleware. V autorském návrhu je middleware naprogramovaný v jazyce Python a běží v cloudové službě Microsoft Azure. Middleware lze provozovat i na vlastním serveru, ale z důvodu, že cloudová služba poskytuje služby typu automatických záloh, škálovatelnost, vysokou dostupnost a automatické přesměrování při výpadku, tak nepředstavuje middleware tak vysoké riziko v rámci SPOF (*single point of failure*) [19].

Tím, že senzory posílají naprosto minimální objem informací (Sigfox odesílá od 1 b po 12 B), implementace standardů (OGC SWE, OGC SimpleThing API) závisí na robustnosti middleware. Je to první stupeň, který je dostatečně energeticky i výkonově zajištěn, aby mohl k přijatým datům připojit i statické informace v podobě metadat, a také odpovídat na standardizované dotazy uživatelů na data a metadata. U NoSQL databází, které obecně nepodporují metodu *JOIN*, jsou naměřená data a metadata vždy

ukládána v jednom dokumentu. U relačních databází lze statická i dynamická data (ať již měněna strojově či manuálně) ukládat odděleně. Za příklad statických dat lze uvažovat seznam použitých senzorů, který může být doplňována o další senzory automaticky pomocí middleware. Například využíváním služeb globální komunitní metadatové databáze všech senzorů a čidel. Existence takové databáze by usnadnila implementace a rozšíření pro veškeré projekty, které by nemusely znovu vytvářet metadatové záznamy o stejných čidlech.

Jedním z dalších problémovým bodem je predikce brzkého začlenění konceptu *Publisher − Subscriber* do stávajících desktop systémů. Z pohledu efektivity aktualizace zobrazovaných dat uživateli v desktopových aplikacích je důležitou vlastnosti aktualizovat obrazová data v okamžiku, kdy jsou na serveru zpracována. QGIS před třemi roky (v roce 2017) představil podporu funkce *Notify/Listen*. U Esri ArcGIS Desktop již nelze čekat další vývoj tímto směrem a to i z důvodu blízkého konce vývoje. ArcGIS Pro je více propojen s webovým prostředím, a lze očekávat, že při vydání Esri ArcGIS Analytics for IoT (vydáno 21. 1. 2020) bude Esri ještě více propojovat web a desktop GIS. U QGIS na obrázku 41 na straně 119 je hned nad funkcí *Notify/Listen* nabídka na nastavení obnovovací frekvence u vrstvy. Jedná se o klasickou metodu, jak nahradit složitější implementaci konceptu *Publisher − Subscriber* v aplikacích fungujících na architektuře *klient − server*. Dotazem na obnovu dat ovšem aplikace odesílá požadavek, i když žádná nová data nejsou k dispozici a tím zvyšují zátěž aplikačního serveru či při přímém připojení i databázového serveru.

7 ZÁVĚR

Cílem této disertační práce byla analýza, návrh a ověření architektury systému umožňující sběr, uložení, zpracování a sdílení velkého množství dat generovaných moderními geograficky rozmístěnými senzorovými sítěmi s využitím distribuovaných prostorových databázových systémů s následným využitím těchto dat v geografických informačních systémech pro tvorbu časoprostorových analýz v reálném čase. Práce předkládá autorský přístup k budování senzorových sítí pomocí LPWAN, předzpracování těchto dat pro využití v GIS a vytváření distribuované databázové sítě pro zvýšení dostupnosti dat. Motivací k disertační práci byla potřeba zpřístupnit proudová senzorová data desktop GIS uživatelům tak, aby bylo možné jejich pokročilé zpracování pomocí časoprostorových analýz v reálném čase. Práce měla ve své experimentální části odpovědět na dvě základní výzkumné otázky (hypotézy):

- Jsou v současnosti dostupné replikační mechanismy v prostorových databázích použitelné pro zabezpečení distribuce prostorových a senzorových dat?
- Je efektivní provést integraci heterogenních senzorových dat do jednotného datového úložiště?

Pro potvrzení či zamítnutí výše uvedených hypotéz byl hlavní cíl práce rozdělen na tři dílčí cíle. První dílčí cíl měl za úkol popsat a otestovat dostupná replikační řešení v návaznosti na jejich funkčnost s prostorovými daty. **Z provedeného testování je výsledkem prvního dílčího cíle potvrzení první hypotézy.** V současné době jsou k dispozici replikační mechanismy v prostorových databázích použitelné pro zabezpečení distribuce prostorových a senzorových dat. Replikace s prostorovými daty je více specifická než replikace klasických datových typů (například číslo či textový řetězec) a mnohdy vyžaduje i specifický přístup. Replikační mechanismy u PostgreSQL s nadstavbou Slony (asynchronní logická replikace) přináší řadu výhod oproti MySQL. Například Slony do-

voluje replikaci jen určitých tabulek (vrstev), replikace je dokončena za mnohem kratší čas (využívá maximální rychlost propojení databázových serverů), který závisí na výkonosti *slave* serveru a počtu lomových bodů v replikovaných datech. Naproti tomu MySQL nezatěžuje tolik CPU a síťové připojení, čas replikace nejvíce ovlivňuje počet replikovaných záznamů, a je zde důležité mít výkonnější *master* server.

Výsledky druhého a třetího dílčího cíle potvrdily druhou hypotézu, že je efektivní provést integraci heterogenních senzorových dat do jednotného datového úložiště. Z ověřených postupů lze efektivně a automatizovaně nastavit celý proces od měření veličin, snížení objemu dat k odeslání, samotné odeslání dat, ošetření hrubých chyb, až po samotné uložení dat do jednotného datového úložiště. Aby byla práce se samotnými daty efektivní, je nutné centralizované datové úložiště distribuovat pomocí replikačních mechanismů, a tím zajistit všem klientům vysokou dostupnost (*high availability*), rozložení zátěže (*load balancing*) a i minimální zálohu dat (i když replikace primárně k zálohování neslouží).

Dalšími výsledky prvního dílčího cíle jsou popisy doporučených forem a struktur používaných distribuovanými prostorovými databázovými systémy pro dosažení distribuovanosti. Dále jsou uvedeny implementační nároky, principy, včetně jednotlivých řešení, které jsou otestovány. Zvláštní zřetel je kladen na replikační a synchronizační mechanismy prostorových databází PostgreSQL a MySQL. Výsledky jsou doprovázeny tabelárními výsledky, grafy a schématy. Následně jsou popsány postupy tvorby distribuované databázové sítě pro konkrétní databázové systémy. Fungování distribuované sítě je popsáno pomocí schémat znázorňujících toky dat v distribuované databázové síti.

Druhý dílčí cíl se zabýval zpracováním senzorových dat produkovaných senzorovými sítěmi komunikujících pomocí technologií Sigfox, LoRaWAN, Zigbee a GPRS. Výsledkem této části jsou zdokumentované metody a formy komunikace mezi senzory, metody odesílání naměřených hodnot pomocí LPWAN sítí (a GPRS) a získávání senzorových dat ze sítě. Závěry z tohoto dílčího cíle byly publikovány v certifikované metodice.

Třetí dílčí cíl si kladl za úkol získaná data očistit a uložit do jednotné prostorové databáze senzorových dat. Získávání dat probíhá pomocí aplikačního middleware, který integruje data z heterogenních senzorových sítí a řeší hrubé chyby v naměřených datech. Middleware je napsán v jazyce Python a běží na cloudové službě Microsoft Azure. Tím, že je middleware napsán v jazyce Python, lze jej provozovat na vlastní nezávislé infrastruktuře. Aby nedošlo k centralizaci dat do fyzicky jediného databázového úložiště, byly využity poznatky z prvního dílčího cíle a byly prakticky otestovány dva možné přístupy vytvoření distribuované databázové sítě. V aplikační rovině byl následně prakticky odzkoušen PostgreSQL nástroj *Notify/Listen*, který vždy s nově přijatými daty do databáze odeslal notifikaci do QGIS, kde byla data automaticky, bez zásahu uživatele, aktualizována a požadovaná *akce*.

Soudobý technologický vývoj, kdy se desktop aplikace upravují a přesouvají na cloudové služby (např. ArcGIS Online) umožňuje využití nových přístupů ke zpracování dat. Proudová senzorová data, která produkují velké objemy dat, tak lze poměrně snadno převést na webové a cloudové služby, a využít tak jejich přidaný potencionál (platba jen za využitý výkon, vysoká dostupnost, automatické zálohy). Desktop GIS má ovšem stále nezastupitelné místo při zpracovávání prostorových dat a proudová senzorová data budou jistě jedním z dalších běžných zdrojů dat. Funkce *Notify/Listen* rozšiřuje klasické fungování desktop GIS z architektury klient − server na koncept *Publisher − Subscriber,* kdy jsou data pasivně přijímána ze zdroje místo aktivního dotazování, zda existují nějaká nová data.

Pro budoucí zpracování velkého množství senzorových dat by mohly být prozkoumány možnosti aktuálně nabízených cloudových služeb, mimo jiné i Esri ArcGIS Analytics for IoT či Esri ArcGIS GeoEvent Server. Dále pak třeba i nasazení open-source integračních nástrojů typu Apache Kafka na vlastní infrastruktuře s přímým napojením na desktop a WebGIS.

Hlavním přínosem disertační práce je ověřená a popsaná architektura systému, která umožňuje získání senzorových dat z různých senzorů pomocí různých komunikačních technologií, uložení senzorových dat v jednotném datovém distribuovaném úložišti a automatické zpracování a využití prostorových dat pro provádění časoprostorových analýz v rámci desktop GIS v reálném čase.

Díky testování byla ověřena schopnost databázových systémů PostgreSQL a MySQL replikovat prostorová data s následným využitím v distribuované databázové síti. Proudová data ze senzorových sítí jsou poskytována do desktop GIS pomocí PostgreSQL funkce *Nofity/Listen* a tak na straně klienta dochází k tvorbě časoprostorových analýz nad aktuálními daty v reálném čase.

Navržená architektura je vhodná pro systémy zpracovávající senzorová data v GIS v reálném čase. Toto téma je velice aktuální, protože je předpoklad, že takovéto architektury budou v budoucnu stále častěji používány, jak poroste počet senzorů, které generují časoprosotorová data.

8 LITERATURA

[1] Abramova, V.; Bernardino, J.; Furtado, P.: Which NoSQL Database? *Open Journal of Databases (OJDB)*, ročník 1, č. 2, 2014: str. 8.

[2] Agarwal, S.; Rajan, K. S.: Performance analysis of MongoDB versus PostGIS/-PostGreSQL databases for line intersection and point containment spatial queries. *Spatial Information Research*, ročník 24, č. 6, dec 2016: s. 671–677, ISSN 2366-3286, doi:10.1007/s41324-016-0059-1.

[3] Aitchison, R. G. F.: An Introduction to DNS. In *Pro DNS and BIND*, Apress, 2005, ISBN 1590594940, s. 3–19, doi:10.1007/978-1-4302-0050-5_1, `arXiv:1011.1669v3`.

[4] Anon.: About IQRF.
Dostupné z: `https://www.iqrf.org/iqrfabout`

[5] Anon.: Data Replication - Data - AI - Analytics Reference Architecture.
Dostupné z: `https://ibm-cloud-architecture.github.io/refarch-data-ai-analytics/data/data-replication/`

[6] Anon.: Internet of Things Global Standards Initiative. 2012.
Dostupné z: `http://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx`

[7] Anon.: Number Of Database Connections - PostgreSQL wiki. mar 2014.
Dostupné z: `https://wiki.postgresql.org/wiki/Number_Of_Database_Connections`

[8] Anon.: 3GPP TR 45.820: Cellular System Support for Ultra Low Complexity and Low Throughput Internet of Things. 2015.

Dostupné z: http://www.3gpp.org/ftp/specs/archive/45_series/45.820/45820-030.zip

[9] Anon.: Data Platforms Map. *451 research*, 2016: str. 1.
Dostupné z: http://info.the451group.com/rs/331-DYY-590/images/MC-2016-Data-Platform-Map-Q1.pdf

[10] Anon.: Iso/Iec 20922:2016. 2016.
Dostupné z: http://www.iso.org/iso/catalogue_detail.htm?csnumber=69466

[11] Anon.: Python Developers Survey 2018 Results. 2018.
Dostupné z: https://www.jetbrains.com/research/python-developers-survey-2018/

[12] Anon: IoT Portál. Technická zpráva, České radiotelekomunikace a.s., Praha, 2019.

[13] Anon.: MySQL. 2019.
Dostupné z: https://www.mysql.com/

[14] Anon.: *MySQL 8.0 Reference Manual (rev. 63056)*, ročník 1. 2019, 6618 s.
Dostupné z: dev.mysql.com

[15] Anon.: PostgreSQL. 2019.
Dostupné z: https://www.postgresql.org/

[16] Anon.: Satcom-IoT via LoRa-WAN - Addvalue Technologies. 2019.
Dostupné z: https://www.addvaluetech.com/satcom-iot-via-lora-wan/

[17] Anon.: SimpleCell - Connecting Things. 2019.
Dostupné z: https://simplecell.eu/

[18] Anon.: solid IT: DB-Engines. 2019.
Dostupné z: http://db-engines.com/en/ranking

[19] Anon: Souhrn smluv o úrovni služeb | Microsoft Azure. feb 2019.
Dostupné z: https://azure.microsoft.com/cs-cz/support/legal/sla/summary/

[20] Anon.: What is New in Zigbee 3.0. Technická zpráva, Texas Instruments, 2019.
Dostupné z: http://www.ti.com/lit/an/swra615a/swra615a.pdf

[21] Anon.: What is Streaming Data? - Amazon Web Services (AWS). 2019.
Dostupné z: https://aws.amazon.com/streaming-data/

[22] Anon.: CAT 5, cat 5e, cat6, cat6a, cat7, cat8 Cable Standards. 2020.
Dostupné z: http://www.cablek.com/technical-reference/cat-5---5e--6--6a---7--8-standards

[23] Anon.: Object oriented DBMS. 2020.
Dostupné z: http://db-engines.com/en/article/Object+oriented+DBMS

[24] Anon.: Roadmap - pgpool Wiki. 2020.
Dostupné z: https://www.pgpool.net/mediawiki/index.php/Roadmap

[25] Anon.: Sigfox Cloud Integration. 2020.
Dostupné z: https://build.sigfox.com/backend-callbacks-and-api

[26] Anon.: Slovník VÚGTK. 2020.
Dostupné z: https://www.vugtk.cz/slovnik/termin.php?jazykova_verze=&tid=1216&l=prostorova-data

[27] ArcGIS Enterprise: GeoEvent Server tutorials - Real-time Data Feeds and Sensors (10.5.1). 2017.
Dostupné z: http://server.arcgis.com/en/geoevent/latest/get-started/geoevent-server-tutorials.htm

[28] ArcGIS Resources: Tracking Server - Welcome to the Esri Tracking Server 10.1 Help. 2015.
Dostupné z: http://resources.arcgis.com/en/help/tracking-server/10.1/index.html#/Welcome_to_the_Esri_Tracking_Server_10_1_Help/00r60000000s000000/

[29] Ayoub, W.; Samhat, A. E.; Nouvel, F.; aj.: Internet of Mobile Things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs Standards and Supported Mobility. *IEEE Communications Surveys and Tutorials*, ročník 21, č. 2, 2019: s. 1561–1581, ISSN 1553877X, doi:10.1109/COMST.2018.2877382.

[30] Baralis, E.; Dalla Valle, A.; Garza, P.; aj.: SQL versus NoSQL databases for geospatial applications. In *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, dec 2017, ISBN 978-1-5386-2715-0, s. 3388–3397, doi:10.1109/BigData.2017.8258324.

[31] Bardyn, J. P.; Melly, T.; Seller, O.; aj.: IoT: The era of LPWAN is starting now. *European Solid-State Circuits Conference*, ročník 2016-Octob, 2016: s. 25–30, ISSN 19308833, doi:10.1109/ESSCIRC.2016.7598235.

[32] Bauer, H.; Veira, J.: The Internet of Things: Sizing up the opportunity | McKinsey & Company. 2014.
Dostupné z: https://www.mckinsey.com/industries/high-tech/our-insights/the-internet-of-things-sizing-up-the-opportunity

[33] Bazydło, P.; Dąbrowski, S.; Szewczyk, R.: Wireless Temperature Measurement System Based on the IQRF Platform. 2015, s. 281–288, doi:10.1007/978-3-319-10990-9_25.

[34] Bell, C.; Kindahl, M.; Thalmann, L.: *MySQL High Availability*. 2014, ISBN 9780596807306, 705 s.

[35] Bennett, T.: NoSQL for GIS Applications. 2015.
Dostupné z: `https://www.linkedin.com/pulse/nosql-gis-applications-thomas-bennett/`

[36] Bereta, K.; Stamoulis, G.; Koubarakis, M.: Ontology-based data access and visualization of big vector and raster data. *International Geoscience and Remote Sensing Symposium (IGARSS)*, ročník 2018-July, 2018: s. 407–410, doi: 10.1109/IGARSS.2018.8518255.

[37] Bereta, K.; Xiao, G.; Koubarakis, M.: Answering geosparql queries over relational data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, ročník 42, č. 4W2, 2017: s. 43–50, ISSN 16821750, doi:10.5194/isprs-archives-XLII-4-W2-43-2017.

[38] Bereta, K.; Xiao, G.; Koubarakis, M.: Ontop-spatial: Ontop of geospatial databases. *Journal of Web Semantics*, ročník 58, 2019: str. 100514, ISSN 15708268, doi:10.1016/j.websem.2019.100514.

[39] Berners-Lee, T.; Hendler, J.; Lissila, O.: The Semantic Web. *Scientific American Magazine*, ročník 284.5, 2001: s. 34–43.
Dostupné z: `ttp://sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21`

[40] Boicea, A.; Radulescu, F.; Agapin, L. I.: MongoDB vs Oracle - Database comparison. *Proceedings - 3rd International Conference on Emerging Intelligent Data and Web Technologies, EIDWT 2012*, 2012: s. 330–335, doi:10.1109/EIDWT.2012.32.

[41] Boswarthick, D.; Elloumi, O.; Hersent, O.: *M2M Communications: A Systems Approach*. John Wiley & Sons, Inc., 2012, ISBN 1119994756, 332 s.

[42] Brizhinev, D.; Toyer, S.; Taylor, K.: Publishing and Using Earth Observation Data with the RDF Data Cube and the Discrete Global Grid System. 2017.
Dostupné z: `https://www.w3.org/TR/eo-qb/`

[43] Brown, E.: 21 Open Source Projects for IoT. 2016.
Dostupné z: `https://www.linux.com/NEWS/21-OPEN-SOURCE-PROJECTS-IOT`

[44] Buttazzo, G.: Real-time operating systems: Problems and novel solutions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, ročník 2469, 2002: s. 37–51, ISSN 16113349.

[45] Campoli, F.: pg_chameleon. 2020.
Dostupné z: `https://pgchameleon.org/`

[46] Carvajal Soto, J. A.; Werner-Kytölä, O.; Jahn, M.; aj.: Towards a Federation of Smart City Services. 2016, doi:10.2991/racs-15.2016.28.

[47] Castell, N.; Dauge, F. R.; Schneider, P.; aj.: Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates? *Environment International*, ročník 99, feb 2017: s. 293–302, ISSN 01604120, doi: 10.1016/j.envint.2016.12.007.

[48] Cattell, R. G. G.; Barry, D. K.; Berler, M.: *The Object Data Standard : ODMG 3.0*. Morgan Kaufmann, 2000, ISBN 1558606475, 280 s.

[49] Charvát, K.; Kocáb, M.; Konečný, M.; aj.: *Geografická data v informační společnosti*. Výzkumný ústav geodetický, topografický a kartografický, v.v.i., 2007, ISBN 978-80-85881-28-8, 280 s.

[50] Chen, Z.; Chen, N.: A real-time and open geographic information system and its application for smart rivers: A case study of the Yangtze River. *ISPRS International Journal of Geo-Information*, ročník 8, č. 3, 2019, ISSN 22209964, doi: 10.3390/ijgi8030114.

[51] Chu, X.: Open Sensor Web Architecture : Core Services. 2005.

[52] Codd, E. F.: A relational model of data for large shared data banks. *Commun. ACM*, ročník 13, č. 6, 1970: s. 377–387, ISSN 0724-6811, doi:10.1145/362384.362685.

[53] Di Gennaro, P.; Lofú, D.; Vitanio, D.; aj.: WaterS: A Sigfox-compliant prototype for water monitoring. *Internet Technology Letters*, ročník 2, č. 1, 2019: str. e74, doi:10.1002/itl2.74.

[54] Dolgikh, M.: *Vysoká dostupnost v relačních databázových systémech*. Diplomová práce, Masarykova univerzita, Brno, 2014.

[55] Ellis, B.: Real-time Analytics: Techniques to Analyze and Visualize Streaming Data. 2014: str. 435.

[56] Erl, T.: *SOA: Servisně orientovaná architektura, kompletní průvodce*. Brno: Computer Press, a.s., 2009, ISBN 978-80-251-1886-3, 671 s.

[57] Filippovová, J.; Pohanka, T.: Environmental Assessment of Central European Floodplain Forests: A Case Study from the Morava River Alluvium. *Polish Journal of Environmental Studies*, ročník 28, č. 6, sep 2019: s. 4511–4517, ISSN 1230-1485, doi:10.15244/pjoes/95041.

[58] Fogel, S.; Morales, T.; Potineni, P.; aj.: *Oracle ®Database Administrator's Guide*, ročník 1. 2008, 882 s.

[59] Francis, B.: Web Thing API. 2019.
Dostupné z: https://iot.mozilla.org/wot/

[60] FraunhoferIOSB: GitHub - FraunhoferIOSB/FROST-Server: A Server implementation of the OGC SensorThings API.
Dostupné z: https://github.com/FraunhoferIOSB/FROST-Server

[61] Fujdiak, R.; Mlynek, P.; Malina, L.; aj.: Development of IQRF technology: Analysis, simulations and experimental measurements. *Elektronika ir Elektrotechnika*, ročník 25, č. 2, 2019: s. 72–79, ISSN 13921215, doi:10.5755/j01.eie.25.2.22739.

[62] Gartner: Gartner Says 6.4 Billion Connected "Things"Will Be in Use in 2016, Up 30 Percent From 2015. 2015.
Dostupné z: http://www.gartner.com/newsroom/id/3165317

[63] Giusto, D.; Iera, A.; Morabito, G.; aj.: *The Internet of Things*. 2010, ISBN 978-1-4419-1673-0, doi:10.1007/978-1-4419-1674-7.

[64] GOST: GitHub - gost/server: GOST - Go implementation of OGC SensorThings API.
Dostupné z: https://github.com/gost/server

[65] Grothe, M.; Carton, L.; Van Den Broecke, J.; aj.: Smart emission - Building a spatial data infrastructure for an environmental citizen sensor network. *CEUR Workshop Proceedings*, ročník 1762, 2016, ISSN 16130073.

[66] Gu, Y.; Wang, X.; Shen, S.; aj.: Analysis of data replication mechanism in NoSQL database MongoDB. *2015 IEEE Int. Conf. Consum. Electron. - Taiwan, ICCE-TW 2015*, 2015: s. 66–67, doi:10.1109/ICCE-TW.2015.7217033.

[67] Guyer, C.; Hubbard, J.; Byham, R.: SQL Server Replication. 2017.

Dostupné z: https://docs.microsoft.com/en-us/sql/relational-databases/replication/sql-server-replication

[68] Haller, A.; Janowicz, K.; Cox, S. J.; aj.: Semantic Sensor Network Ontology. 2017. Dostupné z: https://www.w3.org/TR/vocab-ssn/

[69] Hejlová, V.: *Experimentální bezdrátová senzorová síť pro monitoring znečištění ovzduší ve středu města Olomouce (E-BOSS)*. Dizertační práce, Univerzita Palackého v Olomouci, 2017.

[70] Hejlová, V.; Pohanka, T.; Pechanec, V.; aj.: Communication distance of jennic wireless nodes in the small area. *International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, SGEM*, ročník 1, č. 2, 2015: s. 533–540, ISSN 13142704, doi:10.5593/sgem2015/b21/s8.066.

[71] Hohpe, G.; Woolf, B.: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003, ISBN 0321200683.

[72] Holler, J.; Tsiatsis, V.; Mulligan, C.; aj.: *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence*. 2014, ISBN 9780080994017, 352 s., doi:B978-0-12-407684-6.00001-2.

[73] Huang, C. Y.; Chen, H. H.: An Automatic Embedded Device Registration Procedure Based on the OGC SensorThings API. *Sensors (Basel, Switzerland)*, ročník 19, č. 3, 2019, ISSN 14248220, doi:10.3390/s19030495.

[74] Huang, C. Y.; Wu, C. H.: A web service protocol realizing interoperable internet of things tasking capability. *Sensors (Switzerland)*, ročník 16, č. 9, 2016, ISSN 14248220, doi:10.3390/s16091395.

[75] Huang, W.; Raza, S. A.; Mirzov, O.; aj.: Assessment and Benchmarking of Spatially Enabled RDF Stores for the Next Generation of Spatial Data Infrastructure. *ISPRS International Journal of Geo-Information*, ročník 8, č. 7, 2019: str. 310, doi:10.3390/ijgi8070310.

[76] Inmon, W.; Hackathorn, H.; Richard, D.: *Using the data warehouse*. 1996, ISBN 0471059668, 285 s.
Dostupné z: `https://dl.acm.org/citation.cfm?id=182365`

[77] Janowicz, K.; Haller, A.; Cox, S. J.; aj.: SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, ročník 56, 2019: s. 1–10, ISSN 15708268, doi:10.1016/j.websem.2018.06.003, `arXiv:1805.09979v2`.

[78] Janowicz, K.; Scheider, S.; Stadler, C.; aj.: LinkedGeoData: A Core for a Web of Spatial Open Data. *Semantic Web Journal*, ročník 0, č. 1, 2011, ISSN 1061-5806, doi:10.1080/10615800310001601449.

[79] Jayaram, P.: SQL Server replication: Overview of components and topography. 2018.
Dostupné z: `https://www.sqlshack.com/sql-server-replication-overview-of-components-and-topography/`

[80] Jose A Gutierrez Raymond L Barrett Jr, E. H. C. J.: *Low-Rate Wireless Personal Area Networks: Enabling Wireless Sensors with IEEE 802.15.4*. 2011, ISBN 978-0738162850.

[81] Kabakus, A. T.; Kara, R.: A performance evaluation of in-memory databases. *Journal of King Saud University - Computer and Information Sciences*, ročník 29, č. 4, 2017: s. 520–525, ISSN 22131248, doi:10.1016/j.jksuci.2016.06.007.

[82] Kobialka, T.; Buyya, R.; Deng, P.; aj.: Sensor Web. *Handbook of Research on Developments and Trends in Wireless Sensor Networks*, 2010: s. 447–473, doi: 10.4018/978-1-61520-701-5.ch020.

[83] Kölörčený, M.: Architektury integrace pro rozsáhlé informační systémy a ekonomické aspekty systémové integrace. ročník 13, 2010: s. 161–172, doi:10.7327/cerei.2010.09.05.

[84] Kotsev, A.; Schleidt, K.; Liang, S.; aj.: Extending INSPIRE to the internet of things through sensorthings API. *Geosciences (Switzerland)*, ročník 8, č. 6, 2018: s. 1–22, ISSN 20763263, doi:10.3390/geosciences8060221.

[85] Kovatsch, M.; Matsukura, R.; Lagally, M.; aj.: Web of Things (WoT) Architecture. 2019.
Dostupné z: `https://www.w3.org/TR/wot-architecture/`

[86] Kulkarni, A.: Time-series data simplified | Timescale. 2020.
Dostupné z: `https://www.timescale.com/`

[87] Lake, P.; Crowther, P.: *Concise Guide to Databases*. Undergraduate Topics in Computer Science, London: Springer London, 2013, ISBN 978-1-4471-5600-0, doi: 10.1007/978-1-4471-5601-7.

[88] Lehmann, J.: GeoKnow. 2015.
Dostupné z: `http://geoknow.eu`

[89] Lehmann, J.; Athanasiou, S.; Both, A.; aj.: The geoknow handbook. , č. September, 2015: str. 40.

[90] Liang, S.; Huang, C.; Khalafbeigi, T.: OGC SensorThings API Part 1: Sensing. *Open Geospatial Consortium. Implementation Standard*, 2016: s. 1–105.
Dostupné z: `http://docs.opengeospatial.org/is/15-078r6/15-078r6.html`

[91] Liang, S.; Khalafbeigi, T.: OGC SensorThings API Part 2: Tasking Core. 2019, doi:10.1002/9781118786352.wbieg0348.

[92] Libelium: Waspmote Plug & Sense! Technical Guide. 2016.
Dostupné z: http://www.libelium.com/downloads/documentation/v12/waspmote_plug_and_sense_technical_guide.pdf

[93] Longley, Paul. Goldchild, Mike. Maguire, David. Rhind, D.: *Geographic Information Systems & Science*. Třetí vydání, 2011, ISBN 9780470721445, 539 s.

[94] LoRa Alliance: A technical overview of LoRa and LoRaWAN. 2015.
Dostupné z: https://lora-alliance.org/resource-hub/what-lorawantm

[95] LoRa Alliance Technical Commitee: LoRaWAN 1.1 Specification. 2017.
Dostupné z: https://lora-alliance.org/resource-hub/lorawantm-specification-v11

[96] Makris, A.; Tserpes, K.; Spiliopoulos, G.; aj.: Performance evaluation of MongoDB and PostgreSQL for spatio-temporal data. *CEUR Workshop Proceedings*, ročník 2322, 2019, ISSN 16130073.

[97] Malhotra, N.; Anjali, C.: Implementation of Database Synchronization Technique between Client and Server. *International Journal of Engineering Science and Innovative Technology (IJESIT)*, ročník 3, č. 4, 2014: s. 460–465.

[98] Manyika, J.; Chui, M.; Bisson, P.; aj.: The Internet of Things: Mapping the value beyond the hype. *McKinsey Glob. Inst.*, , č. June, 2015: str. 144, ISSN 1860949X, doi:10.1007/978-3-319-05029-4_7.

[99] Mazilu, M. C.: Database Replication. *Database Syst. J.*, ročník I, č. 2, 2010: s. 33–38, ISSN 21508097.

[100] McGlinn, K.; Blake, D.; O'Sullivan, D.: GViz - An interactive webapp to support GeosparQL over integrated building information. *The Web Conference 2019 - Companion of the World Wide Web Conference, WWW 2019*, 2019: s. 904–912, doi:10.1145/3308560.3316536.

[101] Med, M.; Klímek, J.: Prostorova data. 2019.
Dostupné z: `https://ofn.gov.cz/prostorov{á}-data/2019-08-22/`

[102] Mekki, K.; Bajic, E.; Chaxel, F.; aj.: A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express*, ročník 5, č. 1, 2018: s. 1–7, ISSN 24059595, doi:10.1016/j.icte.2017.12.005.

[103] Milosevic, Z.; Chen, W.; Berry, A.; aj.: Real-Time Analytics. In *Big Data: Principles and Paradigms*, Elsevier, 2016, ISBN 9780128093467, s. 39–61, doi: 10.1016/B978-0-12-805394-2.00002-7.

[104] Nétek, R.; Pohanka, T.: Development of android map application for field collection of agriculture data. *Listy Cukrovarnicke a Reparske*, ročník 135, č. 2, 2019: s. 67–70, ISSN 18059708.

[105] OGC: OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture. *Open Geospatial Consortium, Inc*, 2010: str. 93.

[106] Ondruška, M.: Architektura informačního systému z pohledu integrace v kontextu podnikové integrace. *Systémová Integr.*, ročník 4, 2009: s. 143–158.

[107] Oracle: Anniversary Timeline. 2007: s. 26–33.
Dostupné z: `http://www.oracle.com/us/corporate/profit/p27anniv-timeline-151918.pdf`

[108] Özsu, M. T.; Valduriez, P.: *Principles of distributed database systems*. 2011, ISBN 9781441988331, 1–845 s., doi:10.1007/978-1-4419-8834-8, arXiv:1011.1669v3.

[109] Palmer, M.: IoT arms race is on: Sigfox promises 1bn connections in three years. Dostupné z: https://sifted.eu/articles/sigfox-iot-arms-race/

[110] Pattamsetti, R. M. R.: Distributed Computing in Java 9. 2017: str. 304.

[111] Patterson, M.: Data Stream Processing for Newbies with Kafka, KSQL, and Postgres. 2019.
Dostupné z: https://medium.com/high-alpha/data-stream-processing-for-newbies-with-kafka-ksql-and-postgres-c30309cfaaf8

[112] Pechanec, V.; Machar, I.; Pohanka, T.; aj.: Effectiveness of Natura 2000 system for habitat types protection: A case study from the Czech Republic. *Nature Conservation*, ročník 24, jan 2018: s. 21–41, ISSN 1314-3301, doi:10.3897/natureconservation.24.21608.

[113] Pechanec, V.; Pohanka, T.; Julina, V.; aj.: Integrace obrazových materiálů s daty ze senzorové sítě. Technická zpráva, Univerzita Palackého v Olomouci, Olomouc, 2017.
Dostupné z: http://gislib.upol.cz/moseso/file/metodikaTA04020888-integrace_SS.pdf

[114] Pechanec, V.; Pohanka, T.; Kilianová, H.; aj.: Variability of soil carbon on land borderlines. *Listy Cukrovarnicke a Reparske*, ročník 135, č. 4, 2019: s. 148–151, ISSN 18059708.

[115] Perry, M.; Herring, J.: OGC GeoSPARQL-A geographic query language for RDF data. *OGC Candidate Implementation Standard*, 2012: str. 57.
Dostupné z: http://www.opengis.net/doc/IS/geosparql/1.0

[116] Petäjäjärvi, J.; Mikhaylov, K.; Pettissalo, M.; aj.: Performance of a low-power wide-area network based on lora technology: Doppler robustness, scalability, and coverage. *International Journal of Distributed Sensor Networks*, ročník 13, č. 3, 2017, ISSN 15501477, doi:10.1177/1550147717699412.

[117] Pies, M.; Hajovsky, R.: Use of IQRF technology for detection of construction inclination. 2016, doi:10.1063/1.4952357.

[118] Pohanka, P.: *Architektury výpočetních systémů a zpracování dat v heterogenním síťovém prostředí*. Dizertační práce, Univerzita Obrany, 2013.

[119] Pohanka, P.: Internet of Things standards, consortiums and alliances. 2016. Dostupné z: `http://i2ot.eu/en/blog/internet-of-things-standards-consortiums-and-alliances`

[120] Pohanka, T.; Kilianová, H.; Pechanec, V.: Space-temporal analysis of soil organic characteristics on the ecotone border. *Fresenius Environmental Bulletin*, ročník 28, č. 4A/2019, 2019: s. 3141–3146.

[121] Pohanka, T.; Pechanec, V.; Hejlová, V.: Python Web Server for Sensor Data Visualization. In *16th International Multidisciplinary Scientific Conference SGEM2016, Book 2*, ročník 1, jun 2016, ISBN 978-619-7105-58-2, ISSN 1314-2704, s. 803–810, doi:10.5593/sgem2016B21.

[122] Pohanka, T.; Pechanec, V.; Solanska, M.: Synchronization and replication of geodata in the ESRI platform. *International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, SGEM*, ročník 1, č. 2, 2015: s. 837–843, ISSN 13142704, doi:10.5593/sgem2015/b21/s8.107.

[123] Polakova, M.; Vitols, G.: Use of NoSQL technology for analysis of unstructured spatial data. dec 2018, s. 267–270, doi:10.22616/rrd.24.2018.082.

[124] Postcapes: IoT Standards and Protocols. 2017.
Dostupné z: https://www.postscapes.com/internet-of-things-protocols/

[125] Pour, J.; Gála, L.; Zuzana, Š.: *Podniková Informatika*. Grada Publishing, a.s., 2009, ISBN 978-80-247-2615-1, 496 s.

[126] Rahimi, S. K.; Haug, F. S.: *Distributed Database Management Systems: A Practical Approach*. 2010, ISBN 9780470407455, doi:10.1002/9780470602379.

[127] Redis: Redis Labs Geospatial. 2019.
Dostupné z: https://redislabs.com/redis-best-practices/indexing-patterns/geospatial/

[128] Reinsel, D.; Gantz, J.; Rydning, J.: Data Age 2025: The Digitization of the World From Edge to Core. *International Data Corporation*, , č. November, 2018: str. 28.

[129] Rieke, M.; Bigagli, L.; Herle, S.; aj.: Geospatial IoT - the need for event-driven architectures in contemporary spatial data infrastructures. *ISPRS International Journal of Geo-Information*, ročník 7, č. 10, 2018: s. 1–29, ISSN 22209964, doi: 10.3390/ijgi7100385.

[130] Rooney, N.; Rivoal, F.: World Wide Web Consortium Process Document. 2019.
Dostupné z: https://www.w3.org/2019/Process-20190301/

[131] Samtani, G.: *B2B integration: A practical Guide to Collaborative E-commerce*. World Scientific, 2002, ISBN 1-86094-326-8, 589 s.

[132] Schlienz, J.; Raddino, D.: Narrowband Internet of Things Whitepaper. 2016: str. 42.

[133] Schmid, S.; Galicz, E.; Reinhardt, W.: WMS performance of selected SQL and

NoSQL databases. *ICMT 2015 - Int. Conf. Mil. Technol. 2015*, 2015, doi:10.1109/MILTECHS.2015.7153736.

[134] Schönig, H.-J.: *PostgreSQL Replication*. Packt Publishing, second edi vydání, 2015, ISBN 978-1-78355-060-9, 322 s.

[135] SensorUp: Toronto bike datastreams. 2020.
Dostupné z: `https://toronto-bike-snapshot.sensorup.com/v1.0/Observations(1595467)/Datastream`

[136] Shankland, S.: Oracle buys Sun, becomes hardware company. 2010.
Dostupné z: `https://web.archive.org/web/20100821093146/http://news.cnet.com/8301-30685_3-20000019-264.html`

[137] Shirer, M.; MacGillivray, C.: The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast. jul 2019.
Dostupné z: `https://www.idc.com/getdoc.jsp?containerId=prUS45213219`

[138] Singer, S.: Slony-I. 2020.
Dostupné z: `http://www.slony.info/`

[139] Sinha, R. S.; Wei, Y.; Hwang, S. H.: A survey on LPWA technology: LoRa and NB-IoT. *ICT Express*, ročník 3, č. 1, 2017: s. 14–21, ISSN 24059595, doi:10.1016/j.icte.2017.03.004.

[140] Sohraby, K.; Minoli, D.; Znati, T.: *Wireless Sensor Networks*. 2007, ISBN 9780471743002, 75–92 s., doi:10.1002/9780470112762.ch3.

[141] Solanská, M.: *Synchronizace a replikace geodat v prostředí ESRI platformy*. Magisterská práce, Palackého Univerzita v Olomouci, 2014.

[142] Sumathi, S.; Esakkirajan, S.: *Fundamentals of Relational Database Management Systems*. 2007, ISBN 3540483977, 776 s., doi:10.1007/978-3-540-48399-1.

[143] Sutherland, J.; van den Heuvel, W.-J.: Enterprise application integration and complex adaptive systems. *Communications of the ACM*, ročník 45, č. 10, 2002, ISSN 00010782, doi:10.1145/570907.570932.

[144] Takada, M.: *Distributed systems*. 2013, 35 s.
Dostupné z: `http://book.mixu.net/distsys/`

[145] Trilles, S.; Luján, A.; Belmonte, Ó.; aj.: SEnviro: A sensorized platform proposal using open hardware and open standards. *Sensors (Switzerland)*, ročník 15, č. 3, 2015: s. 5555–5582, ISSN 14248220, doi:10.3390/s150305555.

[146] Trnová, L.: *Hodnocení mechanismů replikace vybraných databázových systémů*. Bakalářská práce, Univerzita Palackého v Olomouci, 2018.

[147] Tschofenig, H.; Arkko, J.; Thaler, D.; aj.: Architectural Considerations in Smart Object Networking. Technická zpráva, mar 2015, doi:10.17487/rfc7452.
Dostupné z: `https://www.rfc-editor.org/info/rfc7452`

[148] Tsiatsis, V.; Höller, J.; Mulligan, C.; aj.: *Internet of Things*. Elsevier, 2019, ISBN 9780128144350, 390 s., doi:10.1016/C2017-0-00369-5.

[149] Uckelmann, D.; Harrison, M.; Michahelles, F.: *Architecting the Internet of Things*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ISBN 978-3-642-19156-5, doi:10.1007/978-3-642-19157-2, `9809069v1`.

[150] Urbano, R.: *Oracle Database Advanced Replication*, ročník 1. Oracle, 2015, 222 s.

[151] Varghese, S. G.; Kurian, C. P.; George, V.; aj.: Comparative study of zigBee topo-

logies for IoT-based lighting automation. *IET Wireless Sensor Systems*, ročník 9, č. 4, 2019: s. 201–207, ISSN 2043-6386, doi:10.1049/iet-wss.2018.5065.

[152] Vrublova, K.; Pohanka, T.: Is Environmental Conservation of European Beech-Dominated Forests Efficient? *Fresenius Environmental Bulletin*, ročník 28, č. 2A, 2019: s. 1218–1223, ISSN 1018-4619.

[153] W3C: Semantic Web - W3C.
Dostupné z: `https://www.w3.org/standards/semanticweb/`

[154] W3C: Semantic Web Standards. 2019.
Dostupné z: `https://www.w3.org/2001/sw/wiki/Main_Page`

[155] Warmerdam, F.; Rouault, E.: GDAL. 2019.
Dostupné z: `https://gdal.org/`

[156] Wibowo, S. B.; Putra, G. D.; Hantono, B. S.: Development of embedded gateway for Wireless Sensor Network and Internet Protocol interoperability. In *2014 6th International Conference on Information Technology and Electrical Engineering (ICITEE)*, IEEE, oct 2014, ISBN 978-1-4799-5303-5, s. 1–4, doi: 10.1109/ICITEED.2014.7007920.

[157] Wingerath, W.; Ritter, N.; Gessert, F.: *Real-Time & Stream Data Management.* SpringerBriefs in Computer Science, Cham: Springer International Publishing, 2019, ISBN 978-3-030-10554-9, doi:10.1007/978-3-030-10555-6.

[158] Wu, C.; Zhu, Q.; Zhang, Y.; aj.: A NoSQL-SQL Hybrid Organization and Management Approach for Real-Time Geospatial Data: A Case Study of Public Security Video Surveillance. *ISPRS International Journal of Geo-Information*, ročník 6, č. 1, jan 2017: str. 21, ISSN 2220-9964, doi:10.3390/ijgi6010021.

[159] Yu, L.: *A Developer's Guide to the Semantic Web*. Berlin, Heidelberg: Springer Berlin Heidelberg, second edi vydání, 2014, ISBN 978-3-662-43795-7, 841 s., doi: 10.1007/978-3-662-43796-4.

[160] Yue, P.; Tan, Z.: *GIS Databases and NoSQL Databases*, ročník 1. Elsevier, 2018, 50–79 s., doi:10.1016/b978-0-12-409548-9.09596-8.

[161] Zhang, X.; Song, W.; Liu, L.: An implementation approach to store GIS spatial data on NoSQL database. *Proc. - 2014 22nd Int. Conf. Geoinformatics, Geoinformatics 2014*, 2014: s. 4–8, ISSN 2161-024X, doi:10.1109/GEOINFORMATICS.2014.6950846.

SUMMARY

The main objective of this doctoral thesis was analysis, design and verification of system architecture, which enables to collect, store, process and share large-scale data generated by modern, geographically distributed wireless sensor networks. Architecture is using distributed spatial database system for the creation of spatiotemporal analyses within GIS in real-time.

The thesis presents author's approach for building wireless sensor networks using LPWAN (Low Power Wide Area Networks) technologies, preprocessing of gathered data for real-time processing within GIS and creating distributed spatial database site to increase data availability. The motivation for the doctoral thesis was a demand to make streaming sensor data accessible in desktop GIS in real-time. Desktop GIS is mostly used for creating advanced spatiotemporal analysis.

The thesis ought to answer two primary research questions:

1. Are there nowadays available replication mechanisms of spatial databases suitable for distribution of spatial and sensor data?

2. Is it efficient to integrate heterogeneous sensor data into unified data storage?

The main objective of the doctoral thesis was split into three sub-objectives to confirm or reject the hypothesis. The first sub-objective aims to describe and test available database replication solutions which provide replication functionality over spatial data. The results from performed testing of the replication mechanisms confirm the first hypothesis. There are nowadays available replication mechanisms of spatial databases for distribution of spatial and sensor data. Spatial data replication is more specific and requires a specific approach than a replication of common data types. PostgreSQL with Slony-I extension and MySQL with build-in replication were chosen for testing. Slony-I allows replication of one or more tables among PostgreSQL servers and replication is

performed in a shorter time than in MySQL. Slony-I replication time depends on the power of a slave server and quantity of edge points in a replicated data. MySQL replication does not require as much CPU power as PostgreSQL and is less demanding on network bandwidth. MySQL replication time is mainly influenced by the number of records and CPU power of a master server.

The results of second and third sub-objective confirm the second hypothesis that integration of heterogeneous sensor data into unified data storage is indeed effective. It is possible to take very efficient and automated steps during the whole process starting from measuring of variables, reducing data size for transfer from the sensor, data transfer itself, solving rough errors in data and ending with storing sensor data into unified data storage. To gain even more efficiency, it is necessary to distribute data storage by replication mechanisms. Distributed storage provides high availability to all clients and load balancing.

Additional results from the first sub-objective are descriptions of recommended forms and structures used for distributed spatial database systems to be distributable. Implementation requirements, principles and solutions for deployment of distributed databases systems are also mentioned.

The second sub-objective aims to processing of sensor data produced by LPWAN sensor networks Sigfox, LoRaWAN, Zigbee and GPRS (not low powered). Results from this sub-objective are well-documented methods and forms of communication among sensors and gates, as well as methods of transmitting measured data by LPWAN and obtaining these data from LPWAN network. Results from this sub-objective are published in certificated methodology.

The third sub-objective aims to purge obtained data from gross errors and storing it into the unified spatial database. Data acquisition from LPWAN networks was performed by application middleware, which integrates data from heterogeneous sensor networks.

Middleware is written in Python, and is deployed on the Microsoft Azure cloud platform. The universality of Python scripting language provides a the possibility of running the middleware on any server or computer.

The architecture was verified also at the application level by testing of PostgreSQL's tool Notify/Listen. This tool is used for notifying QGIS about newly added sensor data into a database. QGIS subsequently automatically performs an action or spatiotemporal analysis over newly received data.

The main contribution of this doctoral thesis is the verified and well-described architecture that allows acquisition of sensor data from different sensors that communicate by different communication technologies. The architecture further allows efficiently integrate and store these heterogeneous data into a unified distributed spatial database that allows performing spatiotemporal analysis by processing sensor data within QGIS.

Efficient spatial data replication of PostgreSQL and MySQL database systems were verified by testing. Spatial data were consequently available within a distributed database site. Streaming sensor data are provided into desktop GIS using PostgreSQL tool "Notify" and Listen function in QGIS for automated creation of advanced spatiotemporal analysis over actual data in real-time.

Architecture designed in this doctoral thesis is suitable for systems that process sensor data within GIS in real-time. This topic is very actual, because there is a premise, that such architectures will be more and more deployed, due to ever-increasing amount of sensors that generate spatiotemporal data.

KATEDRA GEOINFORMATIKY
Univerzita Palackého v Olomouci | Přírodovědecká fakulta

DISTRIBUOVANÉ GEODATABÁZE SENZOROVÝCH DAT
ZÁKLAD PRO INTEGRACI A ANALÝZU

AUTOREFERÁT DISERTAČNÍ PRÁCE
Studijní program: P1314 Geografie
Obor: 1302V011-02 Geoinformatika a kartografie
Školitel: doc. RNDr. Vilém Pechanec, Ph.D.

Mgr. Tomáš Pohanka

DISTRIBUTED SENSOR GEODATABASE
PLATFORM FOR INTEGRATION AND ANALYSIS

Phd THESIS SUMMARY
Study Programme: Geography
Specialization: Geoinformatics and Cartography
Supervisor: Assoc. prof. Vilém Pechanec

Department of Geoinformatics
Faculty of Science, Palacký University Olomouc

Olomouc 2020

Disertační práce byla vypracována v distanční formě doktorského studia na Katedře geoinformatiky Přírodovědecké fakulty Univerzity Palackého v Olomouci.
Dissertation thesis was compiled within Ph.D. study at the Department of Geoinformatics, Faculty of Science, Palacký University Olomouc.

Předkladatel / Submitter:
Mgr. Tomáš Pohanka

Školitel / Supervisor:
doc. RNDr. Vilém Pechanec, Ph.D.
Katedra geoinformatiky
Přírodovědecká fakulta Univerzity Palackého v Olomouci
17. listopadu 50
771 46 Olomouc

Oponenti / Reviewers:
prof. Mgr. Jaroslav Hofierka, PhD. (Univerzita Pavla Jozefa Šafárika v Košiciach)
doc. Ing. Miloslav Hub, Ph.D. (Univerzita Pardubice)
doc. RNDr. Tomáš Řezník, Ph.D. (Masarykova univerzita)

Autoreferát byl rozeslán dne / Summary was posted on: _____

Obhajoba disertační práce se koná dne _____ před komisí pro obhajoby disertačních prací doktorského studia v oboru P1314 Geografie, studijním oboru 1302V011 Geoinformatika a kartografie, v prostorách Katedry geoinformatiky Přírodovědecké fakulty Univerzity Palackého v Olomouci, 17. listopadu 50, 771 46 Olomouc.
The defence of the dissertation thesis will be held on _____ at the commission for the defence of dissertation thesis of Ph.D. degree in study programme P1314 Geography, specialization Geoinformatics and cartography, in the premises of the Department of Geoinformatics, Faculty of Science, Palacký University Olomouc, 17. listopadu 50, 771 46 Olomouc.

S disertační prací je možno se seznámit na studijním oddělení Přírodovědecké fakulty Univerzity Palackého v Olomouci, 17. listopadu 12, 77 46 Olomouc.
The dissertation thesis is available at the Study Department, Faculty of Science, Palacký University in Olomouc, 17. listopadu 12, 771 46 Olomouc.

ISSN 1805-7500
ISBN 978-80-244-5684-3

Obsah

ANOTACE

Disertační práce se věnuje analýze, návrhu a ověření architektury systému, umožňující sběr, uložení, zpracování a sdílení velkého množství dat generovaných moderními geograficky rozmístěnými senzorovými sítěmi s využitím distribuovaných prostorových databázových systémů s následným využitím těchto dat v geografických informačních systémech pro tvorbu časoprostorových analýz. Práce předkládá autorský přístup k budování senzorových sítí pomocí LPWAN, zpracování těchto dat pro využití v GIS a vytváření distribuované databázové sítě pro zvýšení dostupnosti dat. Motivací k disertační práce byla potřeba zpřístupnit proudová senzorová data desktop GIS uživatelům tak, aby bylo možné jejich pokročilé zpracování pomocí časoprostorových analýz. Práce měla ve své experimentální části odpovědět na dvě základní výzkumné otázky (hypotézy):

- Jsou v současnosti dostupné replikační mechanismy v prostorových databázích použitelné pro zabezpečení distribuce prostorových a senzorových dat?
- Je efektivní provést integraci heterogenních senzorových dat do jednotného datového úložiště?

Pro potvrzení či zamítnutí výše uvedených hypotéz byl hlavní cíl práce rozdělen na tři dílčí cíle. První dílčí cíl měl popsat a otestovat dostupná replikační řešení v návaznosti na jejich funkčnost s prostorovými daty. Druhý dílčí cíl se zabýval zpracováním senzorových dat produkovaných senzorovými sítěmi komunikující na technologiích Sigfox, LoRaWAN, Zigbee a GPRS. Třetí dílčí cíl si kladl za úkol získaná data očistit a uložit do jednotné prostorové databáze senzorových dat. Databatáze byla dále distribuována pomocí replikačních mechanismů tak, aby se zamezilo centralizaci dat.

Klíčová slova replikace; prostorová databáze; senzor; komunikace; middleware
Počet normostran textu: 45

1 ÚVOD

Data jsou nedílnou součástí geografických informačních systémů spolu se softwarem, hardwarem a pracovníky, kteří se systémem pracují a využívají ho. Žádná z těchto součástí nesmí chybět, ovšem důležitost jednotlivých součástí se liší. Hardware v dnešní době není nákladný a specializovaný hardware není často nezbytný. Uživatel má možnost výběru software od komerčních poskytovatelů i jako open-source či freeware. Poslední součástí pro geografické informační systémy nezbytné, jsou data. Data mohou být buď zakoupena, využít otevřená data, nebo data získat z vlastních informačních zdrojů, např. terénním sběrem dat, snímky pořízené dronem či automatizovaný sběr dat pomocí senzorů a senzorových sítí [24].

Dle predikce od roku 2015 do roku 2025 vzroste dvojnásobně (z 15 % na 30 %) počet strojově generovaných dat a jejich objem bude dosahovat téměř 80 ZB *(zettabyte)* [37, 39]. I každodenní interakce s digitálním světem se s novými technologiemi (mobilní telefony, hodinky, náramky) výrazně zintenzivňuje a roste produkce nových strojových dat. Úloha přenést tato data do GIS prostředí je nezpochybnitelná. Prvním krokem je produkce vlastních senzorových dat. Dalším pak data zpracovávat, často v reálném čase, ukládat, analyzovat a výsledky z těchto analýz publikovat. Pro prezentaci senzorových dat je nutné zohlednit aspekty (požadavky na) zpracování dat v senzorových sítích, databázových systémech a i v GIS.

Při integraci heterogenních senzorových sítí do centralizovaného databázového úložiště mohou vznikat problémy spojené s celkovým výkonem zařízení. Jedním z možných řešení je centralizované databázové úložiště fyzicky rozdělit ovšem při zachování logické centralizace (z pohledu uživatele). Vytvořením distribuované databázové sítě je zajištěna robustnost celého systému, která inteligentně řeší rozložení zátěže, dostupnost a ochranu dat.

2 CÍL PRÁCE

Cílem této disertační práce je analýza, návrh a ověření architektury systému, umožňující sběr, uložení, zpracování a sdílení velkého množství dat generovaných moderními geograficky rozmístěnými senzorovými sítěmi s využitím distribuovaných prostorových databázových systémů s následným využitím těchto dat v geografických informačních systémech pro tvorbu časoprostorových analýz.

Disertační práce se snaží zodpovědět dvě základní výzkumné otázky (hypotézy):

- Jsou v současnosti dostupné replikační mechanismy v prostorových databázích použitelné pro zabezpečení distribuce prostorových a senzorových dat?
- Je efektivní provést integraci heterogenních senzorových dat do jednotného datového úložiště?

Pro naplnění cíle disertační práce jsou výzkumné aktivity rozděleny na tři logické, na sebe navazujících dílčí cíle. Výsledky a závěry z dílčích cílů v souhrnné podobě slouží pro potvrzení či zamítnutí základních hypotéz. Práce není ovlivněna kvalitou vstupních dat, která velmi souvisí s cenou jednotlivých čidel a účelu nasazení senzorových sítí.

Dílčí cíl 1 – Replikační mechanismy databázových systémů

Prvním dílčím cílem je popis a testování distribuovaných databázových systémů (*DDBS*), jejich vlastností, parametrů a funkcionalit. Definování typů distribuovaných databází a uvedení rozdílů mezi DDBS a distribuovanými prostorovými databázovými systémy (*DPDBS*), včetně podrobného popisu jejich možných přístupů a řešení je nezbytným předpokladem. Primárním předmětem studia distribuovaných (prostorových) databází je forma a způsob využívání replikačních a synchronizačních procesů prostorově orientovaných dat, které jsou zde otestovány a popsány, včetně implementačních nároků jednotlivých řešení DPDBS.

Dílčí cíl 2 – Ověřený postup zpracování dat ze senzorové sítě

Druhý dílčí cíl se zabývá zpracováním dat ze senzorů a senzorových sítí, které mají potenciál stát se jedním z hlavních producentů primárních dat [18, 37]. Jako hlavní zdroj senzorových dat pro potřeby mého výzkumu jsou použity senzory monitorující fyzicko-geografické vlastnosti krajiny. V průběhu práce byla navržena a otestována architektura senzorové sítě komunikující prostřednictvím technologie LoRa, LoRaWAN a Sigfox.

Dílčí cíl 3 – Integrace senzorových dat do jednotné prostorové databáze

Třetí dílčí cíl se zabývá integrací dat spojenou s přenosem dat ze senzorů do jednotné prostorové databáze. Hlavní důraz je kladen na formy, stupně a možnosti integrace dat do jednotné prostorové databáze. Jsou popsány přístupy přenosu dat v reálném čase (*real-time*), proudová data a jejich využití. Pro příjem dat je naprogramován integrační mechanismus, který integruje data přijímaná pomocí technologií LoRaWAN, LoRa, Sigfox, Zigbee a GPRS (*General Packet Radio Service*) do jednotné prostorové databáze. Mechanismus mimo jiné řeší chyby v senzorových datech, například omezení rozsahu hodnot poskytované senzory.

Jako příklad aplikačního využití dosažených výsledků je uvedena architektura pro integraci, přenos, uložení a následnou analýzu v produktu QGIS. Ten jako jediný umožňuje využít funkci *Notify* z PostgreSQL.

3 METODY A POSTUP ZPRACOVÁNÍ

Cíle práce je dosaženo splněním dílčích cílů a potvrzením či zamítnutím hypotéz. V rámci prvního dílčího cíle bylo provedeno testování za účelem určení vhodného způsobu replikace prostorových dat. Druhý dílčí cíl řeší zpracování dat ze senzorových sítí a popisuje komunikace mezi senzorovými sítěmi a senzorovou prostorovou databází. Po zpracování dat jsou poté, v rámci třetího dílčího cíle, pomocí integračního mechanismu (*middleware*) data integrována do jednotné prostorové databáze, která je zároveň distribuována tak, aby se zamezilo datové centralizaci. Následně je zajištěn přenos (primárních či analyzovaných) dat z distribuované databázové sítě do GIS pomocí konceptu *Publisher – Subscriber*. Získané znalosti a podklady vedou k vytvoření architektury celého řešení pro přenos, uložení, zpracování a využití prostorových dat z různých zdrojů v rámci desktop GIS pro možnost provádění časoprostorových analýz.

3.1 Replikační mechanismy databázových systémů

V rámci řešení prvního dílčího cíle byla provedena rešerše z literárních zdrojů týkajících se především prostorových databází a distribuovaných prostorových databází. Základní vlastnosti, výhody a nevýhody jednotlivých komerčních i open-source řešení jsou popsány v podobě textu a tabulek. Aktuální funkcionalita z pohledu databázových i integračních možností, často převyšuje funkcionalitu samotných softwarových řešení geografických informačních systémů. Výběru jednotlivých databázových systémů předcházela rešerše o možnostech integrace databázových systémů a geografických informačních systémů a jejich využitelnosti při návrhu distribuované prostorové databázové sítě.

V praktické části tohoto dílčího cíle bylo prováděno testování a hodnocení replikace prostorových dat. V rámci experimentu byly testovány dva databázové servery – PostgreSQL 9.5 s PostGIS 2.3.3 a MySQL 5.7.19.

PostgreSQL databáze obsahovala rozšíření PostGIS a replikační nástroj Slony-I, který vytváří replikační logiku z hlavní databáze do podřízené databáze (*master – slave replication*). U MySQL databáze bylo využito vestavěné replikační funkce. Testování probíhalo pod mým vedením v rámci bakalářské práce Bc. Lenky Trnové [43].

Testování probíhalo na datových sadách, které jsou běžně dostupné a používané na území České republiky. Jedná se o datové sady ArcČR500 v3.3 (geografická a topografická data pro měřítko 1 : 500 000), Data200 (data pro mapová měřítka 1 : 200 000 vycházející z EuroRegionalMap), NaturalEarth v3.0.1 (data pro mapová měřítka 1 : 10 000 000 obsahující kulturní a fyzické vrstvy i rastrová data pro celá svět) a BPEJ (bonitované půdně-ekologické jednotky) v5.1.2018 pro Olomoucký kraj.

Byla ověřena rychlost samotné synchronizace a identifikována „úzká hrdla“ (*bottleneck*) přenosu dat. Jako zařízení pro testování byl použit standardní osobní počítač a notebook.

Testování probíhalo ve třech režimech propojení serverů:

1. propojení zařízení přes 100 Mbps (*megabit per second*) router,
2. propojení zařízení přes router s omezením šířky pásma na 10 Mbps,
3. přímé propojení zařízení přes 100 Mbps UTP patch kabel kategorie 5E.

3.2 Ověřený postup zpracování dat ze senzorových sítí

Druhý dílčí cíl se zabývá přenosovými technologiemi z oblasti LPWAN, které jsou vhodné pro prostředí IoT a jsou využitelné v České republice a ve světě, včetně jejich přínosu při sběru a zpracování senzorových proudových dat pro GIS.

Pro realizaci experimentů (této části) byly využity senzorové systémy, kterými disponuje Katedra geoinformatiky. Jedná se zejména o patnáct senzorů firmy Libelium. Pro platformu Libelium (základem je Arduino) byly v roce 2017 zakoupeny anténní moduly pro komunikační technologie Sigfox a LoRaWAN. Ty byly vybrány v závislosti na dostupnosti technologie v České republice a pro podporu

projektů TAČR Bezkontaktní monitorování a časoprostorové modelování variability vybraných diferenciačních vlastností půdy (TA04020888) a Systém automatizovaného monitorování a modelování znečištění podzemních vod z nebodových průmyslových zdrojů (TH03030023). Kvalita naměřených veličin a s tím související následná interpretace naměřených dat není v rámci disertační práce řešena. Kvalita naměřených veličin je velmi úzce spjata s cenou čidla a tím i s účelem použití [13].

U technologií Sigfox a LoRaWAN jsou popsány jejich registrační podmínky a způsob připojení do celorepublikové (světové) sítě. Popis je vytvářen ve spolupráci s poskytovateli SimpleCell, respektive Českými Radiotelekomunikacemi (CRa). Dále je u technologie Sigfox diskutována technologie geolokace senzoru bez potřeby globálních družicových polohových systémů. Možnosti využití této služby a přesnost geolokace bude prakticky ověřena.

Dále byly naprogramovány koncové body komunikačních technologií Sigfox Cloud a IoT Portal pro LoRaWAN tak, aby bylo možné pomocí naprogramovaného integračního mechanismu zapisovat data do jednotné prostorové senzorové databáze.

3.3 Integrace senzorových dat do jednotné prostorové databáze

Při řešení třetího dílčího cíle se vycházelo z potřeby integrace heterogenních senzorových sytému do jednotné databáze tak, aby byla tato proudová senzorová data možné dále využívat v GIS systémech. Provedení integrace je nutná pro budoucí využití senzorových dat v desktop GIS, jelikož heterogenita není jen ve vlastnostech senzoru, ale i v datech, která lze ze senzorů získat (z výsledku Ověřený postup zpracování dat ze senzorových sítí). Způsobená centralizace senzorových dat v jednotné prostorové databázi je vyřešena metodami vytvářející distribuované prostředí, zejména replikačními mechanismy (z výsledku Replikační mechanismy databázových systémů).

Naprogramovaný integrační mechanismus, který automaticky rozpozná přijatá data, je součástí webového serveru, který obsluhuje příjem dat, řeší hrubé chyby senzorových dat (hodnoty mimo rozsah měření) a provádí ukládání dat do jednotné prostorové databáze. Webový server i integrační mechanismus je napsán v programovacím jazyce Python. Webový server poskytuje REST API (*Representational State Transfer Application Programming Interface*), které zabezpečuje příjem dat od poskytovatelů (Sigfox a LoRaWAN).

Návrh struktury databáze vycházel z rešeršní části prvního dílčího cíle, ze standardu OGC SensorThing API, z open-source IoT platformy ThingsBoard a z produkční databáze brány Meshlium firmy Libelium. Statická část databáze, která se nemění či jen velmi zřídka, obsahuje metadata o senzorech, čidlech, ale i poloze samotného senzoru. Tyto informace mohou být obsáhlé a rozdělené do jednotlivých tabulek. Návrh dynamické části databáze, která obsahuje výsledky měření čidel senzorů, může být pojata z různých pohledů:

1. každé čidlo každého senzoru má vlastní tabulku,
2. každý senzor má vlastní tabulku se všemi čidly,
3. jediná tabulka pro všechny čidla všech senzorů.

Pro praktické ověření celé architektury distribuované prostorové databáze senzorových dat, byl vytvořen její návrh a byla ověřena její funkčnost. Byl využit desktop GIS QGIS 3.10, který od verze 3 pracuje s PostgreSQL funkcí *Notify/Listen*, která funguje na konceptu *Publisher – Subscriber*.

4 TEORETICKÁ VÝCHODISKA A SOUČASNÝ STAV

Současná infrastruktura prostorových informací budovaná od 90. let 20. století je založena na poskytování dat pouze na žádost (dotaz/odpověď, *request/response*). Stejně tak současný desktop GIS software (ArcGIS, Geomedia, TerrSet, GRASS GIS, QGIS, SAGA GIS, JUMP GIS, gvSIG ...) téměř ve všech případech tuto funkcionalitu podporuje a dokáže s touto logikou pracovat. Historicky byla data velmi těžko dohledatelná, nebyla on-line, byla v proprietárním datovém formátu a chyběla metadata [14]. Dnes již díky vytvořeným standardům a technologickému pokroku není velkou překážkou ani jedno.

Senzorová data mění podobu zpracování dat současnými desktop GIS, které jsou stále nenahraditelné v celém procesu získávání, uložení zpracování a publikování dat. Desktop GIS by měly začít podporovat nové metody jak přistupovat a přijímat data, inovovat kódování a výměny časoprostorových dat z velmi jednoduchých zařízeních (např. v rámci IoT konceptu) a v neposlední řadě přijmout a podporovat nové standardy pro výměnu dat [38].

Oproti architektuře dotaz/odpověď přináší událostmi řízená architektura (*event-driven architecture*) do budoucna podstatné vylepšení celého procesu zpracování proudových prostorových dat. Dnes existují tradiční technologie, které lze použít pro zpracování proudových dat. Jsou jimi například Apache Storm, Apache Kafka, Apache Flink, Apache Spark.

Integrace proudových prostorových dat v desktop GIS v rámci událostmi řízené architektury stále chybí. Pro web a WebGIS, který je mnohem flexibilnější a rychlejší v nasazování nových technologií, je práce s proudovými daty jednodušší a již existují platformy, které proudová data podporují, například ArcGIS GeoEvent Server.

Distribuovaný databázový systém spojuje technologie databázových systémů a technologie počítačových sítí [29]. Hlavní motivací, proč využít databázové systémy je integrace využívaných dat do jednoho místa (centralizace) s následnou kontrolou nad tímto místem. Naopak, počítačová síť jde opačným směrem a snaží se nabídnout decentralizovaný přístup. Hlavní otázkou tedy zůstává, jak tyto dvě technologie mohou vůbec kooperovat. Hlavní myšlenkou distribuovaných databázových systémů tedy není jejich centralizace, ale integrace dat bez jejich centralizace [29]. Distribuovaná databáze se může definovat jako kolekce logicky propojených a fyzicky rozdělených databází v počítačové síti [29, 23].

Databázovou replikací je míněn proces sdílení informací k zajištění konzistence mezi redundantními zdroji (databázemi), zlepšující spolehlivost a přístupnost a snižující chybovost [25]. Databázová replikace vytváří kopie dat, která jsou pak

přístupná z různých míst (jiný server) [44]. Základním stavebním prvkem databázové replikace je uzel (*node*), který reprezentuje jeden databázový server či klastr. Uzel může mít roli nadřízenou (*master*) serveru či podřízenou (*slave*) serveru. Je běžnou praxí používat spojení *master* server a *slave* server v replikačním prostředí.

Obecný pojem „integrace“ můžeme chápat jako propojování počítačových systémů, společností nebo lidí [20, 28]. Počet aplikací a systémů se ve výrobní společnosti ale i v běžné praxi neustále zvyšuje a každá firma si chrání své know-how. Integrace mezi těmito systémy bývá obtížná a zvyšuje finanční i pracovní náročnost na správu a samotnou práci s vyšším množství systémů [42].

Nejběžnější způsoby aplikační integrace lze řešit pomocí základních topologií. Jedná se především o integraci Point to Point, Broker Based, *Publisher – Subscriber*, Message Bus [36]. Integrace senzorových dat byla zvolena pro řešení disertační práce na datové vrstvě a za pomoci databázových replikací. Senzorová data mohou být často integrována až pomocí aplikační vrstvy či vrstvy uživatelského prostředí. Tyto integrace ovšem nejsou vhodné pro další zpracování v prostředí GIS.

4.1 Přenos dat v reálném čase

Podle prof. Buttazza [11] je hlavní rozdíl mezi real-time a non real-time systémem ten, že real-time systém musí úspěšně dokončit úkol v daném časovém intervalu. V real-time systému je výsledek po daném termínu nejen opožděný, ale i nebezpečný.

Real-time označení pro systémy a procesy se zaměřuje kromě správnosti výsledku a správného vyhodnocení pro akční člen také na čas dokončení úkolu. Proudová data (*Stream data či stream analytics*) jsou speciální v tom, že data proudí neustále od většího množství zdrojů a v malých objemech (například v rámci B až kB) [6]. Pokud systém bude zpracovávat data například dvě minuty, ale nová data budou proudit každou minutu, systém za poměrně krátkou dobu přestane být real-time, a nakonec se dostane do fáze, kdy se budou muset některá data zahodit, aby mohl opět plnit svou funkci [16]. Proto proudová data a zpracování proudových dat musí být rychlejší než real-time data. Základní vlastnosti proudových dat je, že proudí neustále z různých zdrojů (např. různé senzorové sítě měřící teplotu), mají volnou a měnící se strukturu a nakonec se databáze musí vypořádat s vysokou kardinalitou dat, tedy s vysokou mírou různých hodnot, které mohou záznamy obsahovat.

4.2 Proudový GIS

V dnešní době se postupně odkláníme od takzvaných pull-based (táhnout) systémů k push-based (tlačit) systémům. Push-based systémy předpokládají, že provedené změny v aplikaci jsou okamžitě zobrazeny ve všech běžících instancích [46]. Jako příklad lze uvést Google Docs, Facebook, komunikátory (*instant messaging* – WhatsApp, Telegram, SMS) či událostmi řízené architektury. Pull-based systémy jsou takové systémy, které pouze odpovídají na dotaz od klienta, například architektura klient – server. Místo zpracování statických a historických dat se přechází na sekvenční zpracování v podobě proudového zpracování a proudových dat (*stream processing, streaming data*).

Proudový GIS rozšiřuje funkcionalitu běžného GIS na GIS, který čte, analyzuje a prezentuje přijatá data průběžně a neustále. Proudové zpracování poté generuje nové výstupy kdykoliv se objeví nová data.

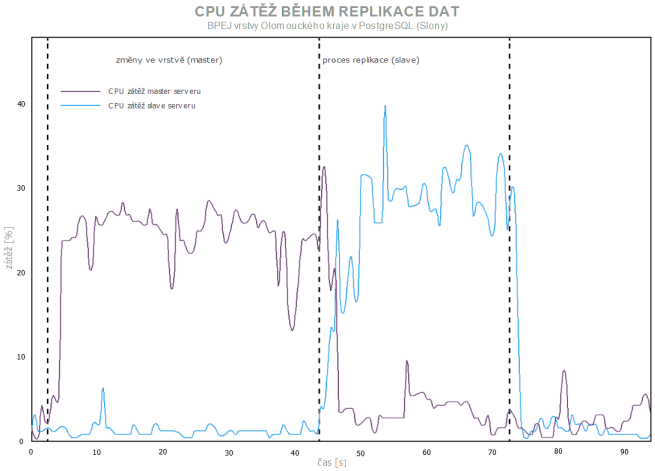
5 VÝSLEDKY

5.1 Testování replikačního mechanismu PostgreSQL a MySQL

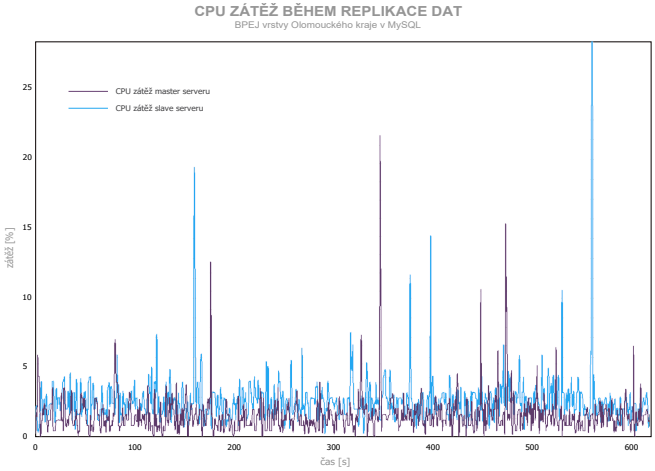
Pro hodnocení replikace prostorových dat byly testovány dva databázové servery. Jednalo se o PostgreSQL 9.5 s PostGIS 2.3.3 a MySQL 5.7.19, které byly nasazena na serveru Katedry. Testování mělo prokázat použitelnost replikačních mechanismů při správě prostorových dat a zjistit úzká hrdla (*bottleneck*) celého procesu. Jelikož úspěšnost replikace prostorových dat byla 100%, bylo měření zaměřeno na vytíženost CPU, přenosovou rychlost a celkový čas replikace jako možných *bottleneck* celého replikačního klastru.

Vytíženost CPU

Na grafech 1 a 2 lze vidět vytíženost CPU (souhrn za všechny jádra CPU) během replikačního procesu BPEJ vrstvy v PostgreSQL a MySQL. Na grafu 1 je jasně patrno, že asynchronní replikace u PostgreSQL nejprve zatěžuje *master* server a až po dokončení změn teprve posílá změny na *slave* server. Naproti tomu MySQL synchronní replikace posílá změny průběžně po jednotlivých záznamech. PostgreSQL mnohem více zatěžuje procesor než MySQL, ovšem v mnohem kratším čase. U PostgreSQL dosahuje průměrného zatížení *master* server přibližně 25 % a *slave* server 30 % zatížení během 40, respektive 30 sekund. To je především způsobeno tím, že *slave* server má výkonnostně slabší CPU, což se projevuje u obou grafů 1 i 2. U MySQL celá synchronizace (grafy 2 a 6 jsou výřezem) trvá až 56× déle (u vrstvy BPEJ), než u PostgreSQL, ovšem s minimálními požadavky na CPU. CPU bude vždy mírně zatížená, například během operačního systému. Proto zátěž do 5 % je možné považovat za téměř klidový stav. Výrazná lokální maxima u MySQL replikace mohou značit vyšší počet lomových bodů záznamu. Celkově však lze konstatovat, že MySQL zatěžuje CPU téměř neznatelně.



Obrázek 1: Graf vytíženosti CPU při replikaci BPEJ vrstvy v PostgreSQL (Slony)

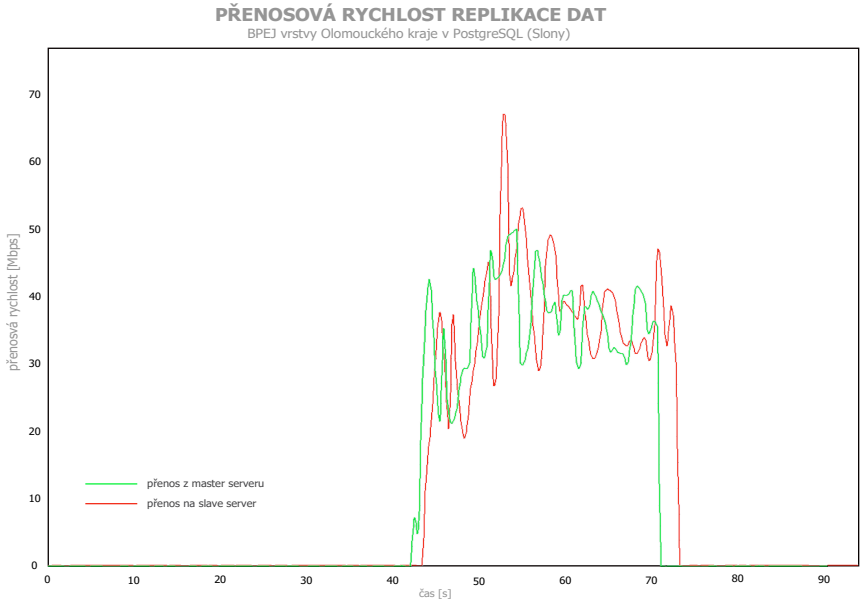


Obrázek 2: Graf vytíženosti CPU při replikaci BPEJ vrstvy v MySQL (výřez)

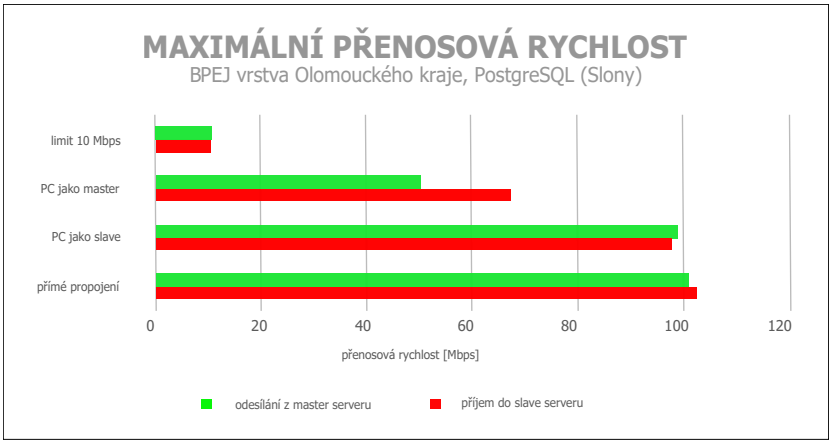
Výsledek sledování zátěže CPU při replikacích jasně ukázal, že PostgreSQL s nadstavbou Slony umožňuje rychlejší replikaci i přes její asynchronní povahu, ale vyžaduje výkonnější CPU. Naproti tomu MySQL využívá synchronní replikace, která je ovšem téměř devětkrát pomalejší, ale nezatěžuje tolik procesor. Nutné je brát zřetel na i na povahu replikovaných dat, tedy polygonová vrstva BPEJ s 3 725 023 lomovými body, 31 280 záznamy o celkové velikosti 228 MB.

Přenosová rychlost

Dalším měřeným kritériem pro hodnocení replikačních mechanismů byla přenosová rychlost dat mezi databázemi v replikačním klastru. Následující grafy 3 a 5 ukazují průběh rychlosti přenášených dat z PC jako *master* server do notebooku jako *slave* server pro PostrgreSQL a MySQL databázi. Propojení serveru v grafech 3 a 5 bylo přes router se 100 Mbps LAN porty. Obecný průběh obou grafů koresponduje s využitím CPU během replikace. I zde se projevuje asynchronicita PostgreSQL a synchronicita MySQL tak, jak se projevila u vytíženosti CPU. V grafu 3 je vidět začátek přenosu dat z *master* serveru a o jednu sekundu později *slave* server začíná data stahovat. I zde, stejně jako při využití CPU, se PostgreSQL snaží využít maximum zdrojů serveru pro co nejrychlejší synchronizaci dat. To samozřejmě klade vyšší požadavky na samotný hardware serveru. Pokud na serveru poběží další služby, které by potřebovaly využívat síť, mohou se navzájem citelně ovlivňovat. Maximální přenosové rychlosti v dalších konfiguracích propojení serverů jsou zobrazeny v grafu 4. PostgreSQL se při jakémkoli typu propojení snaží využít maximální dostupné zdroje. Překvapivé snížení rychlosti při zapojení přes router v konfiguraci *PC jako master* server došlo ke snížení průměrné rychlosti na asi 50 Mbps pro odesílání z *master* serveru. Je to způsobeno pomalejším zápisem *slave* serveru na disk.

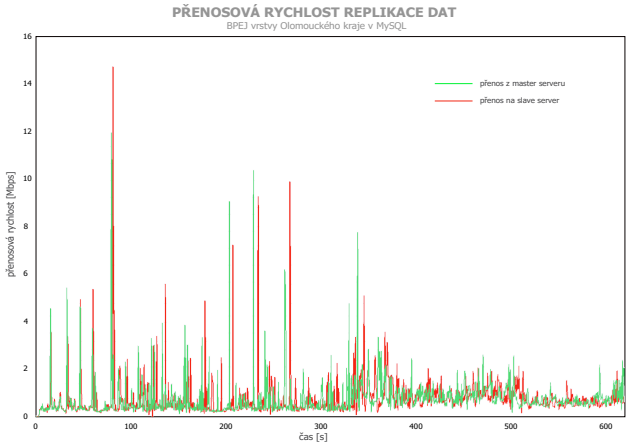


Obrázek 3: Přenosová rychlost replikace dat vrstvy BPEJ v PostgreSQL

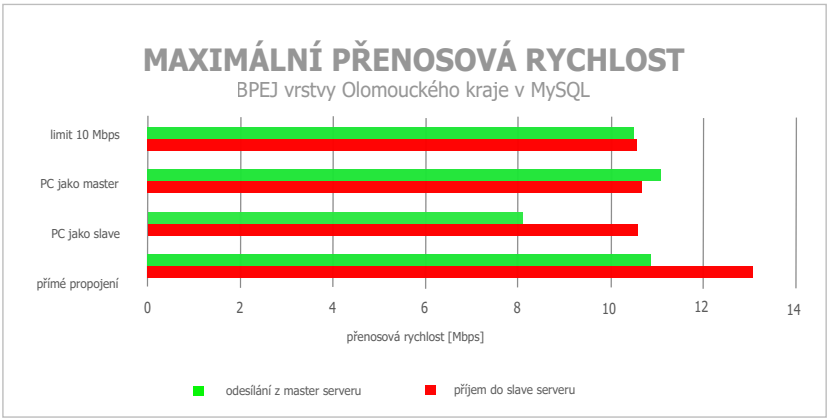


Obrázek 4: Maximální přenosová rychlost replikace dat vrstvy BPEJ v PostgreSQL

U grafu 5 lze pozorovat, že synchronní replikace postupným odesílám jednotlivých záznamů na *slave* server vytěžuje síť jen v jednotkách Mbps. Lokální maxima v průběhu synchronizace značí přenos objemnějších dat (více lomových bodů). Maximální přenosová rychlost (graf 6) ukazuje, že MySQL replikace obecně těsně překročí 10 Mbps. Je to dáno odesílám jednotlivých záznamů okamžitě po jejich změně.



Obrázek 5: Přenosová rychlost replikace dat vrstvy BPEJ v MySQL (výřez)



Obrázek 6: Maximální přenosová rychlost replikace dat vrstvy BPEJ v MySQL

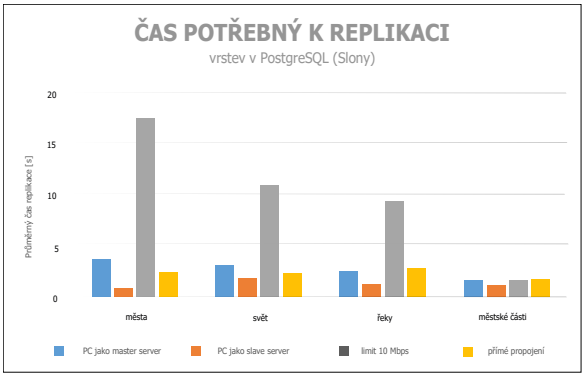
Čas replikace

Mnoho aplikací (práce v týmu, krizový management) závisí na replikaci změn v co nejkratším čase. Následující grafy byly vytvořeny měřením času samotné replikace. Naměřené časy vychází z deseti měření každé změny vrstvy v každé konfiguraci a z výsledných časů byly vytvořeny grafy a tabulka. Již z výsledků při měření vytíženosti CPU a rychlosti přenosu dat je odvoditelné, že celkový čas replikace se bude diametrálně lišit. PostgreSQL s asynchronní replikací synchronizuje změny v BPEJ vrstvě do 29 s, MySQL stejnou operaci při synchronní replikaci zvládne do 1700 s. Hodnoty u všech boxplotů představují medián, 1. a 3. kvartil a extrémní hodnoty.

Obecné srovnání průměrných časů synchronizace vrstev v různých konfiguracích a různých propojení klastru lze vidět na grafech 7 a 8.

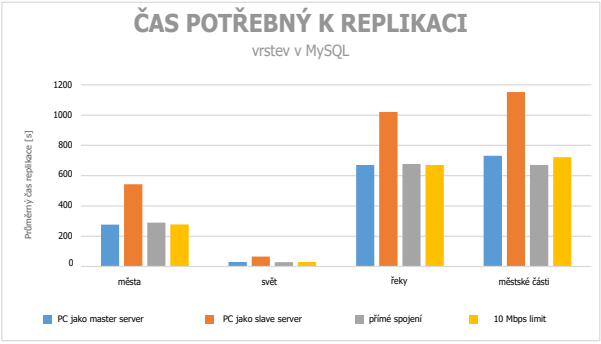
Stabilita replikačních mechanismů byla potvrzena i tím, že všechny časy měly směrodatnou odchylku do 5 % od průměrného času. Pouze u PostgreSQL klastru při konfiguraci *PC jako master* server byla směrodatná odchylka 12 %.

Průměrné replikační časy u PostgreSQL klastru ukazují nejen silnou závislost na dostupné rychlosti připojení, ale také, že s vyšším počtem lomových bodů roste samotná náročnost replikačního procesu. Při konfiguraci s omezenou rychlostí přenosu na 10 Mbps je tento vliv nejvýraznější. Polygonový svět s jediným záznamem, ale s 411 132 lomovými body, je náročnější na replikaci než o poznání objemnější bodové městské části (4 MB vs 12 MB). I při konfiguraci klastru *PC jako master* server lze toto tvrzení potvrdit. Při konfiguraci *PC jako slave* server a při přímém propojení se vyskytují abnormality. Například, že městské části byly synchronizovány rychleji při omezené rychlosti než při přímém zapojení, i když jen o 0,02 s (1,1 %).



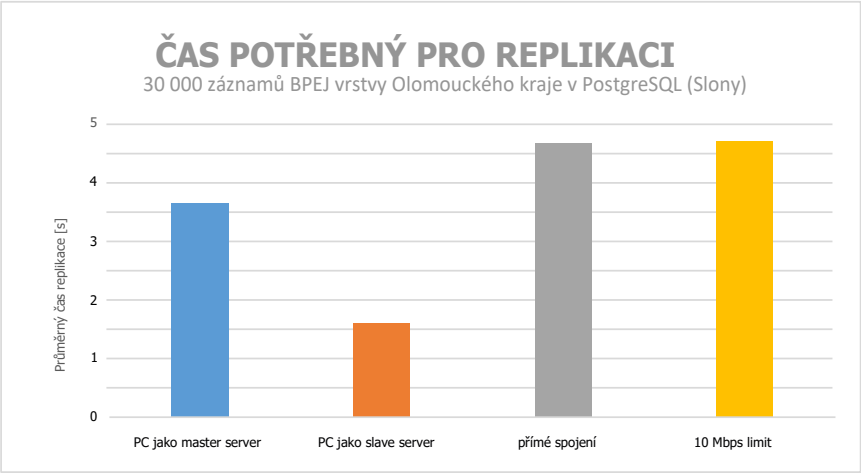
Obrázek 7: Průměrné replikační časy vrstev v různé konfiguraci klastru v PostgreSQL

U MySQL replikace graf průměrného replikačního času (graf 8) naopak ukazuje, že nejvíce ovlivňujícím faktorem je počet synchronizovaných záznamů. Až na konfiguraci s přímým propojením u vrstvy městských částí všechny časy odpovídají tomu, že s vyšším počtem záznamů roste časová náročnost samotné replikace. Řeky a městské části mají velmi podobný počet záznamů, a naprosto rozdílný počet lomových bodu (body vs. linie) a i přesto byly časy velmi podobné (rozdíl při přímém propojení je menší než 1 %). Navíc se ještě projevuje závislost na výkonu *master* serveru, kdy výměna rolí *PC jako master* na *PC jako slave* server měla čas od 45 % až po 65 % horší, než když byl *PC master server*.



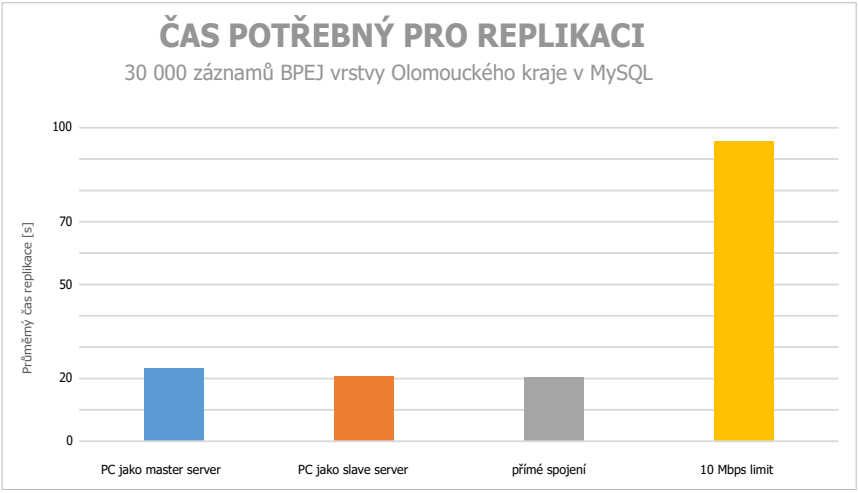
Obrázek 8: Průměrné replikační časy vrstev v různé konfiguraci klastru v MySQL

Testování replikačních časů u negeometrického atributu je další běžnou úlohou při práci s prostorovými daty. Pro zajištění dostatečně dlouhé doby replikace bylo upraveno 30 000 záznamů ve vrstvě BPEJ. I tak změna trvala u PostgreSQL do 5 s (graf 9) a u MySQL do 100 s (graf 10). Zde se také ukazuje vliv velikosti negeometrického a geometrického atributu, kdy negeometrický atribut by při správném návrhu databázového schématu měl respektovat minimálně 3. normální formu. U PostgreSQL (graf 9) se i u negeometrického atributu projevuje závislost výkonu *slave* serveru na celkový čas replikace. Omezení rychlosti v tomto případě nemělo žádný vliv na přenos replikace, i když bylo s přímým propojením nejpomalejší.



Obrázek 9: Průměrný čas replikace atributu u BPEJ vrstvy v různých konfiguracích klastru PostgreSQL

U MySQL se naopak poprvé projevilo omezení rychlosti přenosu na 10 Mbps, které mělo za následek trojnásobné prodloužení času. Dalo by se usuzovat, že MySQL replikace je více optimalizovaná na negeometrické replikace. Zmizel i vliv výkonnějšího zařízení jako *master* server. I tak byl MySQL více než čtyřikrát pomalejší při replikaci než PostgreSQL.



Obrázek 10: Průměrný čas replikace atributu u BPEJ vrstvy v různých konfiguracích klastru MySQL

Testování mělo prokázat použitelnost a využitelnost replikačních mechanismů, jejich úskalí (*bottleneck*) při replikaci prostorových dat. Replikační mechanismy jsou primárně zaměřeny na práci s textovými a numerickými daty. Tomu odpovídají i počty dostupných testů a publikací. Práce s prostorovými daty je více specifická a mnohdy vyžaduje i specifický přístup. Replikační mechanismy u PostgreSQL s nadstavbou Slony (asynchronní logická replikace) přináší řadu výhod oproti MySQL. Například Slony dovoluje replikaci jen určitých tabulek (vrstev). Replikace je dokončena za mnohem kratší čas, který závisí na výkonosti *slave* serveru a počtu lomových bodů v replikovaných datech. Naproti tomu MySQL nezatěžuje tolik CPU a síťové připojení, čas replikace nejvíce ovlivňuje počet replikovaných záznamů, a je důležitější mít výkonnější *master* server.

Splněním prvního dílčího cíle je položen základ architektury distribuované prostorové senzorové databázové sítě. Ta umožňuje zápis, analýzu a poskytování proudových senzorových dat ze senzorových sítí využívající různé komunikační technologie webovým a GIS klientům.

Současně s výše uvedenými výsledky byl replikační mechanismus prakticky využit při řešení v konkrétních výzkumných úlohách [30, 45, 17].

Důvodem pro nasazení replikace pro rozsáhlou prostorovou sadu, byla snaha využít jejího potenciálu pro rozložení zátěže mezi primární databázi, kde dochází k průběžné aktualizaci datové vrstvy (na počítači doc. Pechance) a produkční databázi (můj počítač), která sloužila jako úložiště pro řešení analytické úlohy [35].

Předmětem replikace byla kombinovaná vrstva biotopů ČR. Jedná se o polygonovou vrstvu v rozsahu celé ČR, která v měřítku 1 : 10 000 zaznamenává výskyt a stav jednotlivých biotopů. Vrstva rozlišuje 216 typů biotopů (186 přírodních a 28 nepřírodních) a celkově obsahuje 2 085 099 segmentů. Součástí vrstvy je sada koeficientů pro hodnocení více jak 15-ti vlastností (ekologická hodnota, ekonomická cena, míry plnění vybraných ekosystémových funkcí a služeb). Vlastníkem vrstvy je Ústav výzkumu globální změny Akademie Věd ČR, v.v.i ve spolupráci s Katedrou geoinformatiky Přírodovědecká fakulta Univerzity Palackého v Olomouci a s využitím dat Agentury ochrany přírody a krajiny ČR. Vrstva se využívá k integrovanému modelování a hodnocení stavu krajiny a míry plnění ekosystémových funkcí a služeb.

5.2 Ověřený postup zpracování dat ze senzorové sítě

5.2.1 Automatizované zpracování dat do podoby umožňující bezprostřední využití v analytických úlohách

Hlavním výsledkem tohoto dílčího cíle je vytvoření funkčního a ověřeného automatizovaného zpracování senzorových dat do podoby, která umožňuje bezprostřední využití v analytických úlohách (*preprocessing*). Předmětem řešení bylo stávající „senzorové vybavení“ Katedry geoinformatiky [19]. Aktuální portfólio senzorů a senzorových sítí zahrnuje několik odlišných typů čidel, senzorových desek a podporovaných komunikačních rozhraní. Podrobné informace jsou v tabulce 1, která má návaznost na obrázek 11 na straně 24 a na obrázek 13 na straně 29.

Tabulka 1: Seznam čidel a senzorů

Čidlo	Senzor	Označení v jednotné databázi	Komunikace
Sensiron SHT75	Libelium	HUMB, TCB	Sigfox, LoRaWAN, Zigbee
AMAT 200 mm^2	Libelium, EasyLogGSM	PLV1, PLV2	Sigfox, LoRaWAN, Zigbee, GPRS
Decagon EC-5	Libelium, EasyLogGSM	SOIL1, SOIL2	Sigfox, LoRaWAN, Zigbee, GPRS
DFR0300	Libelium	EC	Sigfox, LoRaWAN, Zigbee
DS18B20	Libelium	TCA	Sigfox, LoRaWAN, Zigbee
HumiAir9	EasyLogGSM	HUMB, TCB	GPRS
Fielder SR03 - 500 mm^2	EasyLogGSM	PLV1	GPRS
Fielder SR02 - 200 mm^2	EasyLogGSM	PLV1	GPRS
Fiedler Virrib	EasyLogGSM	Virrib	GPRS
Baterie	Libelium, EasyLogGSM	BAT	Sigfox, LoRaWAN, Zigbee, GPRS

Tato diverzita jednotlivých řešení a podporovaných komunikačních protokolů způsobuje vysoké nároky na (před)předzpracování vstupních (senzorových) dat při každé úloze. Nutí uživatele mít neustále všechny informace o daném senzoru, způsobu komunikace a o formátu generovaných dat, stejně jako aktivní znalost a neustálý přehled k několika proprietárním programovým řešením pro stažení a kontrolu vstupních dat. Každé technologické řešení rovněž generuje odlišné systémové a nahodilé chyby. Při stávajícím rozsahu senzorové sítě a rychlosti (a objemu) produkce primárních dat je ruční a dílčí *preprocessing* při každém zpracování dále neudržitelné. Předkládané řešení tyto nedostatky odstraňuje a nabízí všem analytikům na pracovišti okamžitý přístup k očištěným a „sjednoceným“ datům.

Dílčí část tohoto postupu byla certifikována v podobě certifikované metodiky č. UKZUZ 020993/2017 Integrace obrazových materiálů s daty ze senzorové sítě [31]. Dále uvedené postupy jsou snadno přenositelné a zobecnitelné na typové řešení např. podle výrobce či komunikačního protokolu.

Následuje popis jednotlivých dílčích kroků zpracovaní dat podle skupin senzorů v závislosti na využívané komunikační technologii.

Senzory komunikující pomocí Sigfox

Senzory komunikující pomocí Sigfox a LoRaWAN technologie posílají data v hexadecimálním zápisu, kde každá hodnota je dvoumístná jednobytová hexadecimální hodnota. Sigfox umožňuje přenášet maximálně 12 B dat v jedné zprávě a maximálně 140× za den. Některé senzorem měřené hodnoty bylo nutno před odesláním upravit do podoby, která by usnadňovala přenos Sigfox a LoRaWAN technologií. U hodnot kapacita baterie v procentech, relativní vlhkost, půdní vlhkost se mohou vyskytovat jen hodnoty od 0 do 100, v hexadecimálním zápisu pak 0 až 64. Hodnoty srážkoměru, které vyjadřují počet impulzů v dané hodině a v předešlé hodině, jsou celými čísly. Ani zde se nepředpokládá, že by hodnota srážek byla nad 255 impulzů za hodinu. Člunek u srážkoměru se záchytnou plochou $200mm^2$ má kapacitu 4 ml na jedno překlopení, které odpovídá $0,2mm$ srážek. V tomto nastavení lze odeslat počet překlopení člunku do úhrnu srážek $50mm$ za 1 hodinu. Pokud by se očekávala vyšší srážka, musí se odesílaná data rozšířit o další byte. U sledování teplot, které mohou nabývat hodnot od -40 do $+80$ (podle provozních hodnot čidla) je nutné odstranit záporná čísla, která by zbytečně zvětšila velikost dat. Ke změřené teplotě se proto přičte hodnota 50. Tím se budou přenášet data z rozsahu od 0 do 100 při teplotách od -50 do $+50$. Měření teploty je navíc poměrně citlivé (odchylka do 3 % podle čidla), a proto se odesílají i desetiny stupně (*rest_ decimal*). Jelikož senzor obsahuje dvě teplotní čidla, je využit rozsah 1 B dat tím způsobem, že hodnota desítek (v decimálním zápisu) je určena jednomu čidlu a hodnota jednotek je určena druhému čidlu. Proměnná *rest_ decimal* bude tedy nabývat hodnot od 00 do 99 (v decimálním zápisu).

Registrace Sigfox zařízení probíhá na základě přidělených kódů ke každému komunikačnímu zařízení. V tomto případě se jedná o Sigfox modul pro Arduino, Waspmote a Raspberry Pi. Každý tento modul má vlastní identifikační kód a tzv. PAC (*Porting Authorization Code*). Oba tyto kódy slouží k identifikaci jednoho modulu. Tyto údaje jsou dodány spolu s modulem. Sigfox modul poté komunikací s bránou vysílá spolu s maximem 12 B uživatelských dat i identifikátor, který

pak síťový server propojí s konkrétním uživatelským účtem v Sigfox Cloud. Spárování Sigfox modulu s Sigfox Cloud poté probíhá čistě přes odeslání první testovací zprávy. Následně je pak uživateli nabídnuto několik způsobů, jak získat data ze Sigfox Cloud. Data lze získat pomocí REST API a Callback API. U REST API po zadání URL adresy a přihlášení, lze získat data ve formátu JSON či YAML. REST API je omezeno na jeden dotaz za sekundu, je proto pro získávání proudových dat nevhodné, ať již z hlediska omezeného počtu dotazů, tak i samotným faktem, že pro získání dat je potřeba odeslání dotazu od klienta. Ten může přijít v době, kdy žádná nová data nebyla přijata, nebo budou přijata těstě po požadavku. U jednoho zařízení lze jednosekundový limit připustit, ovšem u 60 zařízení již dotaz na jedno zařízení bude jen každou minutu. REST API je určeno spíše k administrátorské správě a jednorázovému získání informací o zařízení.

Oproti tomu získávání dat pomocí Callback, které přeposílají data ihned po jejich přijetí (událostmi řízená data, proudová data). Sigfox nabízí jak uživatelský, tak přednastavený Callback. Uživatelský Callback lze nastavit, aby odesílal informace o datech, servisních informací, chybách či událostech. Callback dat může být navíc nastaven tak, aby pouze přeposílal přijatá data na další server nebo může po přeposlání zprávy čekat na data od serveru, která následně pošle do senzoru. Odesílat data do senzoru lze ovšem jen čtyřikrát denně a senzor na ně musí aktivně čekat.

Samotné odesílání dat ze Sigfox Cloud lze na webový server pomocí metod GET, POST, PUT, nebo pomocí emailu. Jako nejpraktičtější se ukázalo odesílání dat pomocí metody POST na webový server. Webový server je hostovaný na platformě Microsoft Azure, který nabízí bezplatný hosting pro Python Web Server (více v kapitole 5.3 na straně 23). Sigfox Cloud odesílá data na webový server pomocí šifrovaného propojení ve formátu JSON.

Vzhled JSON je nakonfigurován přímo v Sigfox Cloud a nabízí předefinované hodnoty, jako například datum, název zařízení či přijatá originální data. Originální data lze přímo v Sigfox Cloudu rozkódovat pomocí uživatelské konfigurace. Ta udává počet bitů a v jaké posloupnosti se ve zprávě vyskytují požadované hodnoty. Tím, že senzor je naprogramován tak, aby odesílal vždy dvoumístné celočíselné číslo (*unsigned integer*) jsou přijatá data v hexadecimálním zápisu rozkódována pomocí konfigurace, kde například *bat* je název proměnné v rámci Sigfox Cloudu a *uint:8* představuje osmi-bitové celé číslo bez znaménka. Po této uživatelské definici přijatých dat lze vytvořit JSON zápis přímo s rozkódovanými uživatelskými daty.

Senzory komunikující pomocí LoRaWAN

LoRaWAN komunikační technologie je po stránce administrace velmi podobná administraci Sigfox. Pro LoRaWAN existuje oficiální poskytovatel (CRa) i soukromí poskytovatelé. Tito soukromí poskytovatelé vlastní LoRaWAN bránu, nejčastěji se jedná o Raspberry Pi s nadstavbou. Tato brána následně přeposílá data cloudovému poskytovateli, například The Things Network, která nabízí propojení soukromých bran do celosvětové sítě tak, aby je mohl zdarma využívat každý uživatel. CRa využívají platformy od firmy Loriot. Největší rozdíl mezi Sigfox a LoRaWAN od CRa je ten, že u LoRaWAN nelze nadefinovat vlastní formát JSON.

Senzory komunikující pomocí GPRS

Dále jsou zapojeny dva dataloggery EasyLogGSM od firmy Physicus. EasyLogGSM je univerzální průmyslový datalogger, který disponuje nízko výkonnostním mikrokontrolorem s kvalitním a přesným analogovým převodníkem. EasyLogGSM umožňuje zpracovávat informace okamžitě díky multiúkolovému operačnímu systému, který dokáže flexibilně a spolehlivě zpracovávat probíhající operace. EasyLogGSM disponuje celkem 12 vstupy. 4 analogové unipolární vstupy (AIN1 – 4) disponují 12-bit převodníkem. Další 4 analogové diferenciální unipolární/bipolární vstupy (A9 – 12) s 24-bit převodníkem. Zbývající 4 vstupy jsou digitální (DIN1 – 4), a jsou naprogramovány tak, aby byly schopny získávat frekvenci (např. rychlost větru), časový interval (sluneční svit) nebo počet impulzů (použití například u člunkového srážkoměru). Každý z 12 vstupů je definován polynomiálními koeficienty (až 3. řádu, $Výstupní\,hodnota = a_0 + a_1x + a_2x^2 + a_3x^3$, kde x je vstupní hodnota a a_0, a_1, a_2, a_3 jsou koeficienty polynomu) měřených hodnot tak, aby mohly být převedeny na odpovídající veličinu (např. z napětí na teplotu ve stupních Celsia). EasyLogGSM komunikuje pomocí dvou sériových portů RS232/RS485. Jeden port je využit na servisní komunikaci a pro nastavení vstupů. Druhý sériový port pak souží pro komunikaci s GSM/GPRS modemem. GSM/GPRS model je schopen odesílat naměřené výsledky v podobě textového souboru na e-mail či na FTP (*File Transport Protocol*) server. EasyLogGSM disponuje také modulem reálných hodin (*real-time clock – RTC*) napájených z baterie typu CR2032. Čas je synchronizován pomocí GPRS jednou denně. Samotný datalogger je napájen šesti kusy baterií typu AA.

Data jsou odesílána na server firmy Ekotechnika, která zajišťuje servis dataloggeru. Data je možné získat třemi způsoby. Na webových stránkách firmy Ekotechnika `http://envirodata.cz/app/` se po přihlášení zobrazí dostupné dataloggery s možností zobrazení dat a manuálního stažení dat. Dalším způsobem je stáhnutí dat pomocí webového API ve formátu JSON. Poslední možností je stáhnout data z SD (*Secure Digital*) karty, kde jsou v jednotlivých souborech (jeden textový soubor za jeden den) uložena data. Datalogger měří hodnoty každou hodinu a odesílá data jednou denně. Tyto časové intervaly byly zvoleny v závislosti na velmi energeticky náročném odesílání dat. Alkalické baterie vydrží tento provoz asi jeden týden, lithiové vydrží půl roku. Jelikož datalogger odesílá data jednou denně, je automatické stahování dat pomocí REST API ideálním řešením. Data z dataloggeru jsou odesílána na speciální email, který spravuje firma Ekotechnika. Data jsou následně ukládána na server, kde je možné k nim přistupovat a stahovat pomocí REST API. Další možností, jak může datalogger odesílat data, je na FTP server. Ovšem využití serverů firmy Ekotechnika, která data zpracuje, zálohuje a publikuje po autorizaci ke stažení pomocí REST API, je z hlediska nákladů a administrace lepší variantou. Více o zpracování dat EasyLogGSM dataloggeru je uvedeno v kapitole 5.3.

V dataloggeru jsou aktuálně připojeny čtyři analogové senzory a jeden digitální. Analogové senzory jsou: Virrib od společnosti Fiedler – elektronika pro ekologii, EC5 od firmy Decagon, HumiAir9 od firmy Ekotechnika a vnitřní teplotní čidlo PT100. Digitální senzor je srážkoměr SR03 od společnosti Fielder – elektronika pro ekologii.

5.2.2 Analytické využití zpracovaných dat

Senzorová data jsou na našem pracovišti využívána v řadě environmentálních studiích. Mezi hlavní aplikační oblasti patří studium vlivu ekotonů na ekosystémové funkce a zejména při studiu variability půdních vlastností v okolí ekotonů a dále modelování retenční schopnosti a proudění vody v krajině. Obě témata mají mimo jiné společný cíl, podporu precizního hospodaření a šetrného využívání přírodních zdrojů.

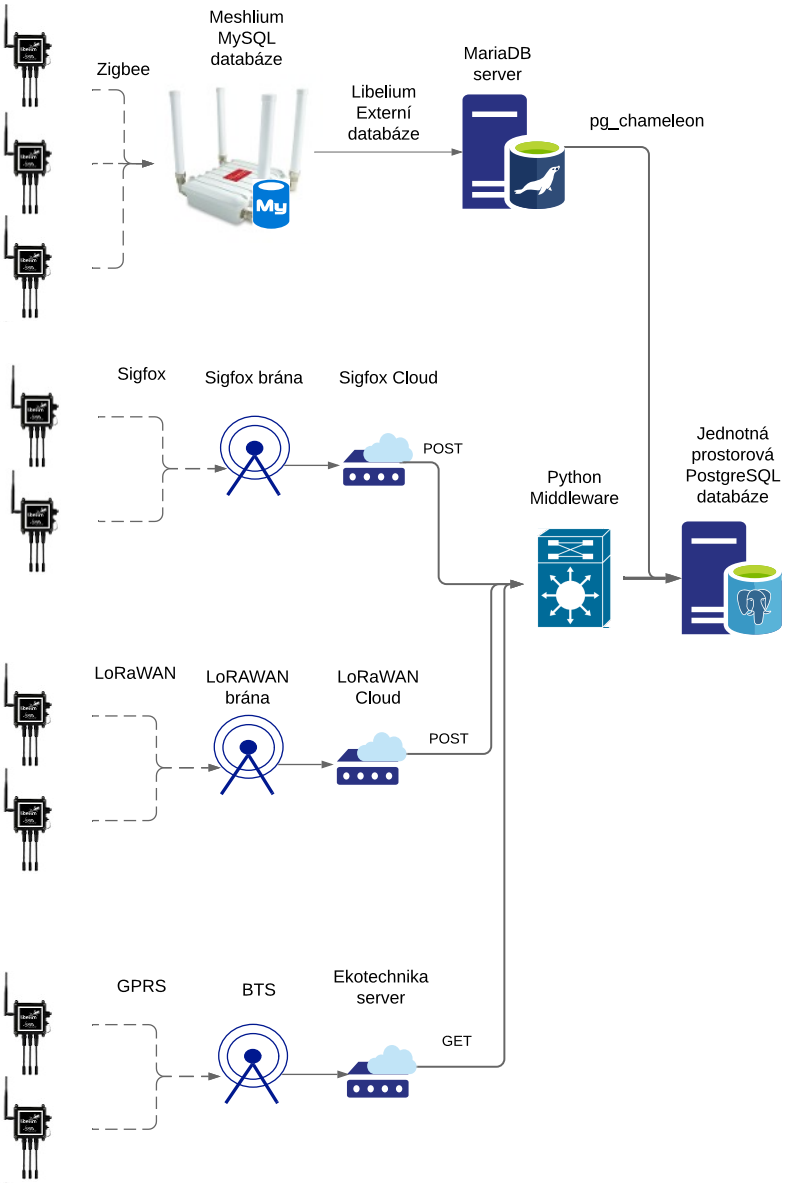
Právě v těchto oblastech je hojně využíváno senzorového měření, které se ukazuje jako vysoce efektivní nástroj pro zajištění dlouhodobého kontinuálního sběru dat v terénu [27].

Realizované studie ([33, 32]) prezentují výsledky studia variability půdního uhlíku. Statistické vyhodnocení potvrzuje, že v blízkosti okraje pozemků kolísají zásoby uhlíku nejen v průběhu roku, ale i v závislosti na vzdálenosti od okraje. Variabilita dostupnosti půdního uhlíku je klíčovým ukazatelem pro agrotechnické opatření.

5.3 Integrace senzorových dat do jednotné prostorové databáze

Výsledkem třetího dílčího cíle je funkční middleware, který zajišťuje integraci senzorových dat do jednotné prostorové databáze. Middleware je naprogramovaný v jazyce Python a běží v cloudové službě Microsoft Azure na webové adrese *https://pohankaiot.azurewebsites.net* Integrační middleware senzorových dat zapisuje veškerá data do jednotné databáze, vzniká tak centrální místo všech dat. „Zpátečnická“ centralizace (vice v kapitole 4 na straně 9) je odstraněna pomocí replikačních mechanismů k vytvoření distribuované databázové sítě.

Python middleware využívá web framework Flask, který se stará o příjem dat. Data jsou následně zpracována moduly *psycopg2* a *json.* Tyto knihovny se starají o parsování přijatých dat ve formátu JSON a o odeslání do jednotné prostorové databáze. Modul *psycopg2* se přímo napojí na databázi a odesílá či přijímá data. Čtení dat z databáze slouží pouze k ověření posledního záznamu v tabulce. Python middleware na obrázku 11 je poslední prvek před jednotnou prostorovou databází.



Obrázek 11: Kompletní schéma přenosu dat ze senzorů do jednotné prostorové PostgreSQL databáze

Jednotná tabulka v databázi slouží k uložení naměřených hodnot jednotlivých čidel senzorů. Data ze senzorů nelze zpětně opravit. Vše řeší Python middleware, který detekuje a vyhodnocuje nepřesná měření a do jednotné databáze tato data vůbec nezaznamená. Pokud senzor vrací nesmyslná data, například vlhkost vzduchu mimo obor hodnot 0 – 100 %, nebo teplotu s hodnotou -1000, pak je jasné, že čidlo je buď chybné, nebo má špatný kontakt. Výsledkem ovšem je, že takto chybně poslaná data nelze „opravit“ na správná a dále s nimi není počítáno.

Metadata konkrétních senzorů a čidel jsou poté navázána na statickou část databáze pomocí *sensor* a *id_sensor*, kde *sensor* atribut obsahuje název (či kód) čidla a *id_sensor* obsahuje název (či kód) senzoru. Poloha jednotlivých čidel je ve speciální tabulce *sensor_loc*. Rozsah tabulky *sensor_loc* závisí na povaze senzoru, např. pro zemědělství a krajinu bude senzor delší dobu na jednom místě. Pro sledování vozidel (zejména komunikační technologie GPRS, NB-IoT, LTE-M) bude počet záznamů v tabulce více záviset na frekvenci odesílání nových dat. Lze i uvažovat, že změnu polohy může provést i jednotlivé čidlo senzoru, například, pokud se čidlo vymění za kalibrované.

Data do jednotné databáze proudí aktuálně ze tří různých zdrojů a to ze senzorů Libelium se Zigbee a Sigfox (LoRaWAN) technologií a z dataloggerů EasyLogGSM. Z dataloggerů EasyLogGSM jsou odesílána data pomocí GPRS jednou denně, vždy po půlnoci, aby byla daná dostatečná časová rezerva pro zpracování dat firmou Ekotechnika. Middleware je nastaven tak, aby odesílal dotaz na nová data každý den vždy v 7:00. Na straně middleware dochází ke kontrole poslední zapsané hodnoty v databázi. Pokud by došlo k jakékoliv chybě, jak na straně middleware či na straně serveru firmy Ekotechnika, budou data stažena za celé období výpadku. Zároveň to řeší i problematiku výskytu duplicitních hodnot.

Data se získávají po přihlášení autorizovaného uživatele přes REST metodou GET, ve které se určí, za jaké období pro který senzor a pro která čidla. Seznam senzorů a odpovídajících čidel lze také získat dotazem na server firmy Ekotechnika, který opět vrací JSON. Následné vytvoření dotazu na konkrétní data lze buď manuálně, nebo automaticky parsováním JSON dat z výsledku dotazu na senzory a čidla. Atributy GET metody jsou: a) časové rozmezí; b) označení senzoru a čidla („Loc428-Col1“). Middleware následně každý den získá data ze serveru firmy Ekotechnika v JSON formátu. Vloží naměřenou hodnotu pro každý senzor přímo do jednotné prostorové databáze senzorových dat.

Ze senzorů Libelium jsou data odesílána třemi způsoby, a to pomocí technologií Sigfox, LoRaWAN a Zigbee. Senzory jsou nastaveny tak, aby odesílaly naměřené informace okamžitě po jejich naměření, které je provedeno každých 15 minut. Délka měření závislí na použitých senzorech, např. senzor Sensiron SHT75 má 10 s kalibrační interval, po který je senzor napájen, a po těchto 10 s se teprve měří hodnota teploty a vlhkosti vzduchu. Senzory Libelium disponují RTC modulem, který dokáže zařízení každých 15 minut probudit a vykonat naprogramovanou úlohu. Data jsou přes síť Sigfox odeslána na Sigfox Cloud, odkud jsou pomocí Callback přeposílány na zabezpečený (*SSL*) webový server s middleware. Webový server je nasazen na platformě Microsoft Azure. Ten nabízí zdarma základní prostor pro vytvoření zejména webových stránek. Webový server lze ovšem naprogramovat pomocí Python jazyka, a lze tak využít výpočetní výkon serveru i k příjmu, odeslání a zpracování

dat. Služba je omezena časovým využíváním CPU na 60 minut za den a 1 GB RAM (*Random Access Memory*). Aktuálně jsou na server posílány dotazy od dvou senzorů přes Sigfox Cloud a je využito asi tři minuty CPU času a 70 MB RAM (obrázek). Nebyl by tedy problém obsluhovat na zdarma poskytované platformě i 30 senzorů, i když v omezené míře. Middleware, který běží na Microsoft Azure, lze takto provozovat na jakémkoli vlastním serveru, čistě s podporou Python jazyka. Hlavní část middleware, který se stará o příjem, zpracování a uložení přijatých dat ze Sigfox Cloud, je zobrazen v algoritmu 1. Middleware využívá webový framework Flask, který je spolu s Django jedním z nejpoužívanějších [3].

Algoritmus 1 Část Python middleware pro příjem, zpracování a odeslání dat

```
@app.route('/', methods = ['POST'])
def home():
 if request.headers['Content-Type'] == 'application/json':
  if request.headers['auth'] == 'disertacniprace2020':
   with psycopg2.connect("dbname='sensor' user='**********' host
       =158.194.94.*** password='**********' ") as con:
    cur = con.cursor()
    cur.execute("INSERT INTO public.sigfox (value) VALUES ('%s')"
        % (json.dumps(request.json)),)
    myjson = request.json
    time = myjson['time']
    device = myjson['device']
    rest_s = 0
    rest_t = 0
    for sensor in myjson:
     if sensor == 'rest':
      rest_s = str(myjson[sensor])[0]
      rest_t = str(myjson[sensor])[1]
      for sensor in myjson:
       if sensor not in ['time', 'device', 'data', 'rest']:
        # v~sensoru přidávám 50 aby -50 stupnu byla 0 a~50 stupnů
            byla 100
        value = myjson[sensor]
        if sensor == 'TCA':
            if value > 0 and value] < 100:
              value = float(value) - 50 + float(rest_t)*0.1
        elif sensor == 'TCB':
          if value > 0 and value < 100:
            value = float(value) - 50 + float(rest_s)*0.1
      cur.execute("INSERT INTO public.data (id_sensor, sensor,
          value, time) VALUES ('%s','%s', %.2f, '%s')" % (device
          , sensor, float(value), datetime.fromtimestamp(int(
          time)).strftime('%d-%m-%Y %H:%M:%S')),)
return render_template('index.html', title='Home Page', year=
    datetime.now().year, data = "sended")
```

Dekorátor @app.route('/', methods = ['POST']) určuje, že přímo na doménovém jménu nejvyššího řádu (například .cz, .com) může přijímat data pouze metodou POST. Vzápětí následuje první podmínka na data, zda mají v hlavičce uvedeno,

že se jedná o JSON data. Následuje připojení k databázovému serveru, který běží na Katedře geoinformatiky (KGI). Jako absolutní záloha, jsou data takto přijatá uložena do tabulky *sigfox* v databázi *sensor*. Tato tabulka obsahuje pouze identifikátor (datový typ *serial*) a data (datový typ *jsonb*). PostgreSQL umí pracovat s dokumenty JSON a XML přímo. Proto lze vytvářet dotazy i přímo na tento JSON dokument. JSON ovšem slouží jako záloha primárních nezpracovaných dat a dotazy na JSON objekt jsou proto mírně pomalejší, než dotazy na klasickou tabulku. Aby bylo zamezeno tzv. SQL Injection, tedy podvrhnutí škodlivého kódu, Sigfox Cloud umožnuje do šifrované hlavičky zadat i autentizační uživatelské údaje. Kód dále rozloží JSON dokument na jednotlivé elementy, které zpracuje a uloží do jednotné tabulky *data* pro všechny senzory.

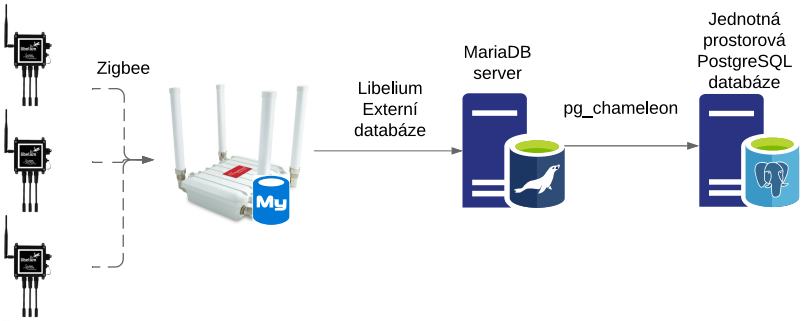
Dalším integračním prvkem, který je nutné řešit je zpracování dat ze senzorů Libelium, jež komunikují pomocí technologie Zigbee. Senzory komunikující na technologii Zigbee komunikují s bránou. U firmy Libelium se brána jmenuje Meshlium a je postavena na operačním systému Debian a databázi MySQL 5.0.51a. Senzorová data jsou ukládána do tabulky *sensorParser*, Nejdůležitější pole jsou *id* s automaticky navyšovaným identifikačním číslem, *id_wasp* s označením konkrétního senzoru, *sensor* s označením konkrétního čidla, *value* s naměřenou hodnotou (v textovém formátu), *timesptamp* s časovou značkou přijmutí dat a nakonec *sync*, které označuje, zda byla data odeslána na externí server.

Odeslání dat na externí server je velmi vítaná funkce přímo implementovaná v bráně Meshlium. Externí databáze ovšem může být pouze MySQL nebo MariaDB, a tabulka musí mít shodnou strukturu. Hlavní databáze v Meshlium pak jen přiřazuje příznak v atributu *sync*, zda byl záznam úspěšně odeslán na externí server. Jako externí server byl použit jeden z počítačů Katedry geoinformatiky. Na serveru je nasazen balíček XAMPP (*Cross-Platform, Apache, MariaDB, PHP a Perl*) ve verzi 5.6.28, který instaluje a propojuje Apache HTTP Server, MariaDB a PHP. Na tento server jsou následně odesílána data z Meshlium. Důvod vzniku tohoto serveru a přeposílání dat má několik důvodů. Jedná se zejména o: a) záloha primárních dat; b) rozložení zátěže; c) bezpečnost z hlediska přístupů do databáze.

Záloha primárních nezpracovaných dat by měla být součástí každého projektu. Nejhorší situace nastane, pokud celý senzor zkolabuje (nejčastěji vybitá baterie). Chybu čidla lze vypozorovat z přijatých dat. Pokud není dostupná brána pro příjem dat, lze data zpětně načíst z vnitřní paměti senzoru. Při nedostupnosti externího serveru budou data automaticky přeposlána z Meshlium brány (podle atributu *sync*). Každá havárie se dá ošetřit upozorněním na nečekanou událost, například v databázovém triggeru nebo middleware.

Architektura senzorů komunikujících na technologii Zigbee od firmy Libelium synchronizuje data do jednotné databáze bez účasti naprogramovaného integračního middleware. Tím, že jednotná prostorová databáze senzorových dat využívá PostgreSQL a externí databáze Meshlium brány je MariaDB, je využit nástroj *pg_chameleon*. Ten vytváří replikační spojení MariaDB a PostgreSQL. Replika je nastavena tak, že PostgreSQL je *slave* server MariaDB serveru. Jedná se o asynchronní logickou replikaci. Výkon nástroje a celkové zdržení při replikaci je u asi 2 000 nově vložených záznamů za sekundu do MariaDB znamenalo asi 10 s zpoždění v PostgreSQL [12]. Schéma toku dat je zobrazeno na obrázku 12. Pg_chameleon

dokáže replikovat jednotlivé tabulky z databáze. Nelze ovšem definovat konkrétní atributy tabulky. Je tedy replikována celá tabulka. Tím, že jednotná prostorová senzorová databáze nevychází přímo z Libelium Meshlium databáze, jsou data ze senzorů uložena nejdříve do samostatné tabulky, a následně pomocí funkce vkládána do jednotné tabulky. Tato modifikace struktury může probíhat na MariaDB serveru, tak na PostgreSQL. Funkce při každém novém přidaném záznamu do tabulky z MariaDB (pomocí spouště (*trigger*)) ověří minimálně, že data jsou v zadaném rozsahu hodnot, a poté je uloží do jednotné tabulky.



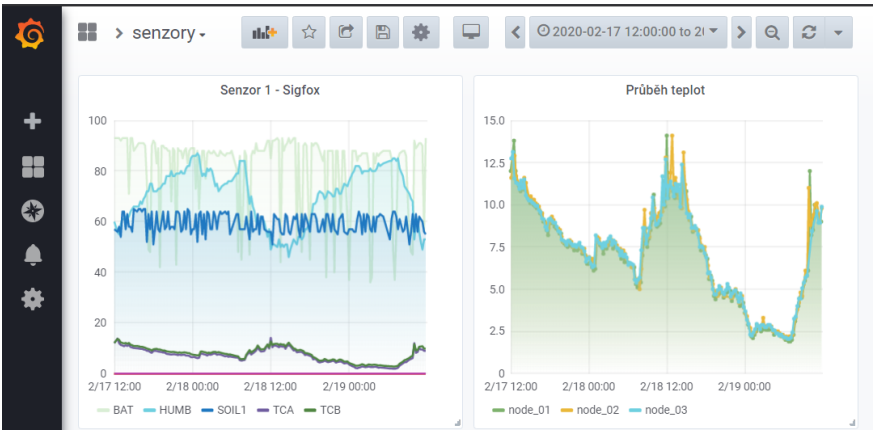
Obrázek 12: Schéma propojení senzorů se Zigbee komunikační technologií s jednotnou PostgreSQL databází

S rostoucím počtem senzorů a délkou sběru dat vyvstává otázka ohledně dělení tabulky (*table partitioning*). Obecným předpokladem pro vytváření částí tabulky je, že velikost tabulky by neměla přesáhnout velikost RAM. Jediná tabulka může mít například v PostgreSQL až 128 TB a neomezený počet řádků. I hodnota 128 TB dat v jedné tabulce může být za určitý čas dosažena. Aby bylo možné tento limit obejít, lze vytvářet části tabulky, kde bude mít každá část tabulky určité velikostní omezení. Ať již se jedná o velikost počtu sloupců (vertikální členění), typ záznamů (historické a aktuální záznamy; horizontální členění), nebo statický (např. 1 000 000 řádků) či dynamický (týdenní části) počet záznamů či využití vertikálního, horizontálního a dynamického členění zároveň. Tohoto mixu využívá například rozšíření TimescaleDB pro PostgreSQL [22]. Části tabulky se ovšem zobrazují a chovají jako by byla tabulka pouze jedna. O dynamické slučování pro zobrazování se stará interní systém databázového serveru.

Dělení tabulky by se mělo opírat i o reálné využití dat v praxi. Zejména dotazy na data budou podstatně rychlejší, pokud bude většina dotazů z jedné části. Například, pokud velmi často směřují dotazy na data za jeden měsíc, je vhodné nastavit dělení tabulky po měsících tak, aby byly dotazy směrovány do jednoho celku. I dotazy na týdenní části tabulky budou řádově rychlejší než dotaz na nedělenou tabulku se stovkami milionů záznamů a velikostí několik desítek GB. Mimo zvýšení

propustnosti čtení dat je zvýšen i samotný zápis dat. Data se připisují jen k části dat, a ta v určitý moment vytvoří další část. Nevýhodou je samozřejmě vyšší režie na administraci databáze. Další nevýhodou je při dělení tabulky nutné využít takový typ replikace, u které je možné replikovat i změnu schématu.

Pro vizuální představu lze data jednoduše zobrazit do grafu. Lze například využít webovou aplikaci Grafana, která nabízí přímé napojení na PostgreSQL. Tím lze veškerá data přehledně zobrazit v grafu (obrázek 13). Grafana dovoluje nastavit krok aktualizace, ale jedná se o klasický model klient – server. Když do databáze budou přidána nová data, Grafana tato data nezobrazí dokud uživatel nepošle požadavek na aktualizaci. Grafana data pouze zobrazuje, žádné analýzy zde provést nelze. Dalším možným zobrazením dat i s využitím prostorových informací je řešen v mé studii [34].



Obrázek 13: Zobrazení naměřených dat pomocí senzorů v grafech.

5.3.1 Distribuování jednotné prostorové databáze senzorových dat

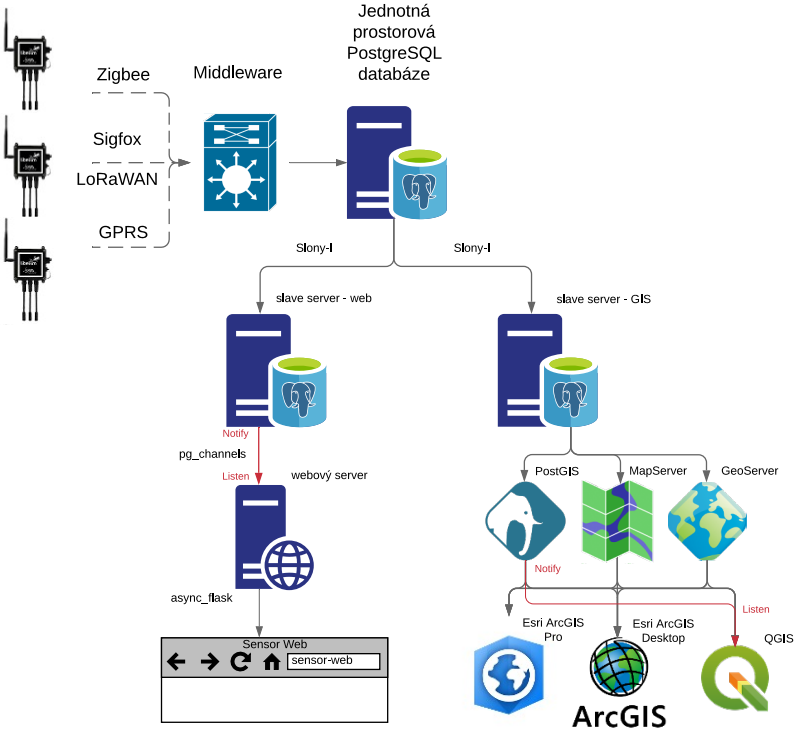
Integrace všech senzorových dat do jednotné prostorové databáze vedlo k její centralizaci. Pro zachování vysoké dostupnosti, optimální výkonosti a minimální zálohu dat, je vhodné centralizovanou databázi distribuovat. Pro PostgreSQL existuje mnoho možností, jak distribuovat databázi. Dva prakticky realizované přístupy, z teoreticky několika možných řešení, jsou popsány níže.

Varianta 1 – s využitím rozšíření Slony-I

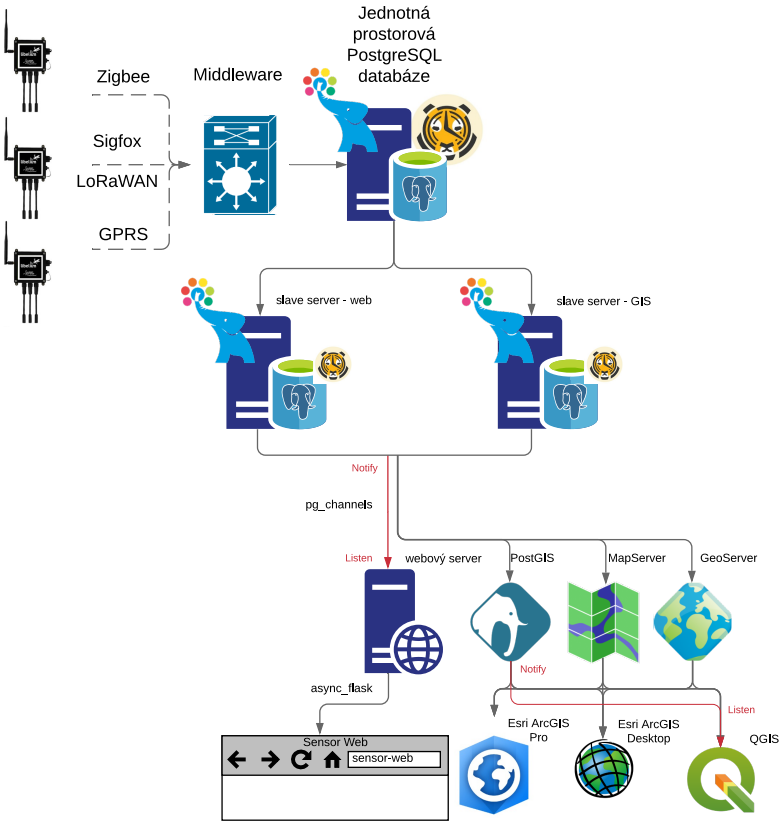
Hlavním záměrem bylo vytvoření funkční cesty ze senzorů až po QGIS využívající PostgreSQL funkce *Notify/Listen*.

První vytvořený přístup využívá Slony-I replikačního systému pro vytvoření replikačního klastru. Jednotná prostorová databáze, která je plněna senzory komunikující na různých technologiích a která vychází z výsledků dílčích cílů je ve schématu 14 využívána jako *master* databáze. Z této databáze jsou následně vytvořeny dvě

asynchronní repliky. Dvě repliky proto, aby mohli být odděleni weboví a desktop uživatelé, a tím se ještě více rozložila zátěž. Pro webového uživatele nemusí být databázový server tak výkonný. Naopak, pro desktop GIS klienty je vhodnější použít výkonný server. Rozlišení databáze pro web i pro GIS je na základě přidělené IP adresy serveru. Pro web a GIS *slave* servery je rozhodující jejich schopnost číst a odesílat data. Tento přístup je vhodný, pokud se počet záznamů pohybuje v desítkách milionů až nižších jednotek stovek milionů v závislosti na velikosti volné RAM serveru. V navržené struktuře jednotné prostorové databáze představuje 1 000 000 záznamů asi 150 MB. U serveru s 16 GB RAM je u počtu 100 000 000 záznamů, tedy asi 15 GB dat v jedné tabulce hranicí, kdy by se podle doporučení měla vytvořit první část tabulky při horizontální dělení. 100 000 000 záznamů se při počtu 1 000 senzorů, které budou posílat data každých 30 minut dosáhne za 5,7 let.



Obrázek 14: První přístup distribuovaného prostředí



Obrázek 15: Druhý přístup distribuovaného prostředí

Varianta 2 – s využitím rozšíření PgPool a TimescaleDB

U druhého přístupu byla provedena inicializace PgPool, bylo přidáno rozšíření TimescaleDB a vyzkoušena replikace mezi *master* a *slave* databází. Druhý návrh vychází ze získaných zkušeností a dokumentace k PgPool a TimescaleDB.

Tento přístup využívá databázové dělení do částí pomocí TimescaleDB i PostgreSQL streaming replikaci a middleware pro vyrovnání zátěže a celkový management replikačního klastru PgPool-II. Při této konfiguraci již téměř nezáleží na počtu senzorů posílající data. TimescaleDB vytváří části tabulky přesně podle požadavků s přihlédnutím na způsob využití dat (čtení poslední hodnoty, čtení hodnot z posledního týdne, měsíce) i s přihlédnutím na hardware serveru, zejména pak na velikost RAM.

Pokud budou uvažovány stejné hodnoty jako u prvního přístupu, pak pro jeden oddíl na jeden měsíc při uvažované velikosti 30 GB (asi 200 000 000 záznamů) a při odesílání dat senzorem každých 10 minut (nejvyšší počet zpráv za den pro LPWAN) by bylo potřeba 46 300 senzorů. TimescaleDB by za jeden měsíc vytvořil další horizontální oddíl, a díky neomezené velikosti PostgreSQL databáze lze tyto oddíly vytvářet neustále. PgPool běží na každém serveru z důvodu pokrytí výpadku jedné z databáze. Při výpadku *master* databáze se okamžitě povýší jeden ze *slave* serverů a nedojde tak k žádnému výpadku. Jakmile bude původní *master* databáze opět funkční, synchronizují se provedené změny z aktuální *master* databáze a z původní *master* databáze se stane *slave* databáze. Klient (Python middleware, QGIS, GeoServer ...) tuto výměnu vůbec nepocítí, jelikož s databázovým klastrem komunikuje pomocí virtuální IP adresy pro celý klastr. PgPool pak sám rozhodne, na kterou z databází klienta připojí. Pokud přijde požadavek na zápis, přepošle data na *master* databázi (jednotná prostorová PostgreSQL databáze či právě aktuální *master* databáze), pokud přijde dotaz na čtení, využije se jeden ze *slave* serverů. Jakýkoli uživatel, ať již s požadavkem na zápis či čtení, se připojuje pouze k jedné IP adrese.

5.3.2 Aplikační využití

Pro vytvoření funkční cesty od senzorů až po QGIS byla využita funkcionalita *Notify/Listen,* poprvé představena v QGIS 3 (vydána 24. 2. 2018). Jedná se o nativní funkcionalitu PostgreSQL od verze 9 (vydána 20. 9. 2010), která není závislá na žádném dalším modulu. V PostgreSQL databázi je nutné vytvořit funkci a spouštěč, která bude zachytávat změny v tabulce *data* a následně funkcí odesílat zprávu o nových, změněných, smazaných záznamech.

Tím je na straně databázového serveru vytvořen (publikován) kanál pro odběr (*subscription*) zpráv. Kdykoliv se v tabulce *data* jakkoli změní data, odešle se zprávě kanálem *qgis*. Na straně QGIS je nutné se připojit k PostGIS databázi. U prvního přístupu distribuovaného prostředí (obrázek 14) je to databáze *Slave server - GIS*, u druhého přístupu (obrázek 15) se QGIS připojí k virtuální IP adrese. Po přidání požadované vrstvy do QGIS, se u těchto dat nastaví odběr kanálu *qgis* (Vlastnosti vrstvy – Vykreslování – Obnovit vrstvu při oznámení).

Nastavení odběru zajišťuje, že při změně dat v jednotné prostorové databázi senzorových dat se projeví změna i v QGIS bez uživatelské interakce. Díky možnosti nastavení a spuštění vlastní *akce* na základě notifikace lze takto vytvářet plně automatické analýzy. Makra (Vlastnosti vrstvy – Akce (Makro)) lze vytvářet pro obecný systém, ale i pro Windows, MacOS, Linux či čistě jen otevřít soubor (například obrázek). Makro ovšem lze především naprogramovat v Python, takže jakákoliv úloha, která lze vytvořit v QGIS může být zde naprogramována. Vytvořený ukázkový algoritmus (2) vytváří Obalovou zónu (*buffer*) okolo bodů. Algoritmus by samozřejmě mohl vytvářet i interpolace a další pokročilé analytické úlohy. Zde se jedná pouze o ukázku možné dalšího využití tohoto nástroje.

Algoritmus 2 Algoritmus makra pro vytváření obalové zóny

```
from qgis import processing
project = QgsProject.instance()

# vymaž starou vrstvu z~předešlé notifikace
for buff in project.mapLayersByName('Obalová zóna'):
  project.removeMapLayer(buff.id())

#vytvoř Obalovou zónu
result = processing.runAndLoadResults("native:buffer",
            {'INPUT': 'data',
              'DISTANCE': 0.1,
              'SEGMENTS': 10,
              'DISSOLVE': True,
              'END_CAP_STYLE': 0,
              'JOIN_STYLE': 0,
              'MITER_LIMIT': 10,
              'OUTPUT': 'memory:test'})
result_layer= result["OUTPUT"]

#nechť je vrstva viditelná
project.layerTreeRoot().findLayer(result_layer).
    setItemVisibilityChecked(True)
```

6 DISKUZE

V průběhu tvorby práce muselo docházet k řadě rozhodnutí, kterou cestou se výzkum bude ubírat. Bylo provedeno rozhodnutí o výběru technologií, komponent, testovaných algoritmech apod. Z toho důvodu lze předložený výzkum a navržené řešení označit jako autorský přístup, který není jediný možný, ale je zvolen s nejlepším vědomím a svědomím a na základě dostupných znalostí a prostředků. Dosažené výsledky a ověřený provoz jsou však důkazem, že výzkum se ubíral správným směrem.

Výsledky prvního dílčího cíle jsou zaměřeny na replikační možnosti databázových systémů PostgreSQL a MySQL. Výběr databázových systémů podléhal zvoleným kritériím, kde velkou váhu má využití databázových systému na KGI, ale i snadné rozšíření odporné komunitě pracující s nízkonákladovými senzory a nízkonákladovou, nízkoenergetickou komunikační technologií. Výběr replikačních mechanismů pro PostgreSQL a MySQL vycházel z rešeršní části a i z prací vytvořených na KGI (práce Mgr. M. Dobré (rozená Solanská) [41] a Bc. L. Trnové [43]). U MySQL lze využít jen vestavěné funkce, kdežto u PostgreSQL lze využít mnoho existujících rozšířeních, kdy každé z nich poskytuje rozdílné metody replikace. Slony i PgPool se velmi často objevoval v odborných publikacích a tématických diskusích a neustálý vývoj nástrojů zaručuje jejich podporu i v nejnovějších verzích PostgreSQL [40, 8].

Dalším významným rozhodnutím bylo omezení přenosové rychlosti na 10 Mbps. To bylo zvoleno s ohledem na dnešní technologie, například IEEE 802.11b z roku 1999 komunikovalo 11 Mbps či HSPA+ (*High Speed Packet Access*), někdy označováno jako 3,5G komunikovalo právě rychlostí 10 Mbps. Dnes již lze

komunikovat rychlostmi například u 5G rychlostí až 10 Gbps (*Gigabit per second*), stávající WiFi6 (ve dřívějším označení 802.11ax) má teoretický limit při 9,6 Gbps a budoucí WiFi7 (802.11be) až 30 Gbps. UTP patch kabel kategorie 5 má limit 100 Mbps, 5E má 1 Gbps, kategorie 6 a 7 má maximum 10 Gbps a připravovaná kategorie 8 má maximum při 40 Gbps [7]. 10 Mbps je z dnešního pohledu tedy extrémně nízká rychlost, který byla zvolena jen pro tento test. Omezení rychlosti má simulovat reálný provoz serveru, kdy na jednom zařízení běží více softwarových serverů či služeb. Typicky na jediném hardwarovém serveru jsou souběžně spuštěny webový, databázový i aplikační server, které využívají sdíleného internetového připojení. V rámci předtestování bylo také zjištěno, že velký vliv na celou replikaci má centrální procesorová jednotka (*CPU*) [15]. Proto byla měřena nejen vytíženost síťového připojení ale i vytíženost CPU. Pro měření a omezení rychlosti připojení byl využit program NetLimiter 4, pro měření vytíženosti CPU byl využit HWMonitor Pro v1.3.

Hodnocení replikačního klastru se opíralo o stávající metody hodnocení výkonu databázového serveru jako jsou TPC-x (*Trasaction Processing Performance Council*), SSB (*Star Schema Benchmark*) či YCSB (*Yahoo! Cloud Services Benchmark*) [1, 10, 21, 15]. Tyto hodnotící testy databáze zjišťují například maximální počet uložení za sekundu, čas odezvy při určitém počtu uživatelů či čtecí a editační testy. Ty se ovšem zaměřují převážně na číselné a textové datové typy.

Výhody distribuovaných (prostorových) databází se v malém měřítku téměř neprojeví a spíše převáží nevýhody ve formě vyšší náročnosti na spuštění a celkovou administraci distribuované sítě. Ale už při jednotkách souběžných připojení na jedinou databázi může být server ovlivněn minimálně zvýšenou odezvu [2]. Replikační databázové sítě (klastry) jsou nejvíce vidět u mezinárodních firem, kde mezikontinentální zpoždění v komunikaci je již značné. Geo-replikace (replikace s cílem snížit komunikační zpoždění na velkých vzdálenostech) jak jí označuje Microsoft Azure či Cross-Region replikace u Amazon S3, využívají zejména firmy, které nabízejí cloudové služby. Pro služby typu Microsoft Azure, Amazon AWS, Maptiler Cloud, se již bez geo-replikace neobejdou.

Výsledky druhého dílčího cíle měly vyřešit problematiku heterogenních senzorových sítí s cílem integrace dat do jednotné prostorové databáze. Senzory byly úmyslně osazeny rozdílnými komunikačními technologiemi, aby se zvýšila testovaná heterogenita a tím se zvýšila potřeba naprogramování vlastního integračního middleware (výsledek třetího dílčího cíle). Komunikační technologie jsou z oblasti LPWAN (až na GPRS), které umožňují komunikaci na velkou vzdálenost s velmi malou energetickou náročností. Mají i své nevýhody, jako například omezené pokrytí signálem, v některých případech nízké časové rozlišení či odesílání dat v objemech jednotkách B [26]. Představují však možné řešení problému s využitím senzorových měření i v místech, kde není zajištěn pevný zdroj energie, například v lesích, polích či horách.

Míra komprimace přenášených dat byla před odesláním ze senzorů upravena tak, aby každé čidlo odesílalo celočíselnou hodnotu v rozsahu 0 – 255 (1 B v hexadecimálním zápisu 0 – FF). Je to z důvodu využití parsování dat na straně Sigfox Cloud a odeslat JSON již s konkrétními čidly s jejich naměřenými hodnotami. V samotném senzoru mohla být naprogramována vyšší komprimace dat například pomocí

LZW či RLE. Data by se pak místo na cloudu služby Sigfox či LoRaWAN parsovala až v samotném middleware, kde by se hexadecimální řetězec dekomprimoval a až následně by se hodnoty přiřadily konkrétním čidlům.

Odesílání dat je možné řešit několika způsoby a byl vybrán ten, který je možné využít jak pro službu Sigfox tak pro LoRaWAN. Bylo zvoleno odesálání dat ve formátu JSON pomocí metody POST na SSL zabezpečený server. U Sigfox Cloud je možné si vytvořit vlastní strukturu JSON, kdežto z LoRaWAN IoT Portálu lze data odesílat pouze v nadefinovaném formátu [4]. Pro příjem zpráv z uživatelského *Callback* lze vybrat ze dvou možností, a to jako *jednoduchá*, nebo *pokročilá* data. *Pokročilá* data na rozdíl od *jednoduchých* podporují: a) výpočet polohy senzoru na základě síly přijatého signálu bránami; b) fixní polohu zadanou u metadat senzoru; c) kód země ve které se senzor nachází. Vypočítaná poloha má podle síly přijatého signálu přesnost řádově jednotky km v závislosti na počtu bran, které zprávu přijaly. *Pokročilá* data mohou být až o 30 s zpožděná oproti *jednoduchým* datům. Fixní poloha senzoru může být změněna přímo na administrační stránce v Sigfox Cloud nebo pomocí REST API přes metodu PUT. Využití vypočtené polohy je tedy velmi omezené a lze ji využít všude tam, kde přesnost i několik jednotek kilometrů nehraje roli, např. pro hrubé sledování zásilky, průběžné sledování mezinárodní dopravy.

Pokud uživatel zná polohu senzoru, je možné ji přímo zadat do Sigfox Cloud. Změny polohy následně pomocí REST API nebo ručně na webové stránce vložit přímo ke každému senzoru a využívat příjem pokročilých dat ze Sigfox Cloud i se zadanou polohou. Pokud ovšem uživatel má informaci o aktuální poloze, lze ji i přímo zapsat do databáze, a tím zkrátit prodlevu přijetí zprávy ze Sigfox Cloud, snížit datový tok a nepřijímat duplicitní, nebo neměnné informace o poloze.

Sigfox Cloud nabízí i přednastavou funkci *Callback,* která existuje pro AWS IoT (*Amazon Web Services*) a Kinesis. Dále pak pro Microsoft Azure IoT Hub a Event Hub a i pro IBM Watson IoT Platform [9]. Zmíněné PaaS cloudové služby mohou přijímat miliony zpráv za sekundu pomocí HTML, MQTT, AMQT atp. protokolů, jsou škálovatelné, bez složitých konfigurací a bez administrace vlastního serveru. Výkon je ovšem za poplatek od jednotek korun za hodinu až po několik tisíc.

Při zpracovávání pokročilých časoprostorových analýz se často využívají data z různých zdrojů s různým měřítkem či prostorovým rozlišením. U strojových senzorových dat lze integraci vyřešit i právě pomocí middleware. V mém návrhu je middleware naprogramovaný v jazyce Python a běží v cloudové službě Microsoft Azure. Middleware lze provozovat i na vlastním serveru, ale z důvodu, že cloudová služba poskytuje služby typu automatických záloh, škálovatelnost, vysokou dostupnost a automatické přesměrování při výpadku, tak nepředstavuje middleware tak vysoké riziko v rámci SPOF (*single point of failure*) [5].

Tím, že senzory posílají naprosto minimální objem informací (Sigfox odesílá od 1 b po 12 B), implementace standardů (OGC SWE, OGC SimpleThing API) závisí na robustnosti middleware. Je to první stupeň, který je dostatečně energeticky i výkonově zajištěn, aby mohl k přijatým datům připojit i statické informace v podobě metadat, a také odpovídat na standardizované dotazy uživatelů na data a metadata. U NoSQL databází, které obecně nepodporují metodu *JOIN*, jsou naměřená data a metadata vždy ukládána v jednom dokumentu. U relačních databází

lze statická i dynamická data (ať již měněna strojově či manuálně) ukládat odděleně. Za příklad statických dat lze uvažovat seznam použitých senzorů, který může být doplňována o další senzory automaticky pomocí middleware. Například využíváním služeb globální komunitní metadatové databáze všech senzorů a čidel. Existence takové databáze by usnadnila implementace a rozšíření pro veškeré projekty, které by nemusely znovu vytvářet metadatové záznamy o stejných čidlech.

Asi posledním problémovým bode je predikce brzkého začlenění konceptu *Publisher – Subscriber* do stávajících desktop systémů. QGIS před třemi roky představil svojí podporu funkci *Notify/Listen*. U Esri ArcGIS Desktop již nelze čekat další vývoj tímto směrem a to i z důvodu blízkého konce vývoje. ArcGIS Pro je více propojen s webovým prostředím, a lze očekávat, že při vydání Esri ArcGIS Analytics for IoT (vydáno 21. 1. 2020) bude Esri ještě více propojovat web a desktop GIS. U QGIS je hned nad funkcí *Notify/Listen* nabídka na nastavení obnovovací frekvence u vrstvy. Jedná se o klasickou metodu, jak nahradit složitější implementaci konceptu *Publisher – Subscriber* v aplikacích fungující na architektuře *klient – server*. Dotaz na obnovu dat ovšem aplikace odesílá požadavek, i když žádná nová data nejsou a tím zvyšují zátěž aplikačního či při přímém připojení i databázového serveru.

7 ZÁVĚR

Cílem této disertační práce byla analýza, návrh a ověření architektury systému umožňující sběr, uložení, zpracování a sdílení velkého množství dat generovaných moderními geograficky rozmístěnými senzorovými sítěmi s využitím distribuovaných prostorových databázových systémů s následným využitím těchto dat v geografických informačních systémech pro tvorbu časoprostorových analýz v reálném čase. Práce předkládá autorský přístup k budování senzorových sítí pomocí LPWAN, předzpracování těchto dat pro využití v GIS a vytváření distribuované databázové sítě pro zvýšení dostupnosti dat. Motivací k disertační práci byla potřeba zpřístupnit proudová senzorová data desktop GIS uživatelům tak, aby bylo možné jejich pokročilé zpracování pomocí časoprostorových analýz v reálném čase. Práce měla ve své experimentální části odpovědět na dvě základní výzkumné otázky (hypotézy):

- Jsou v současnosti dostupné replikační mechanismy v prostorových databázích použitelné pro zabezpečení distribuce prostorových a senzorových dat?
- Je efektivní provést integraci heterogenních senzorových dat do jednotného datového úložiště?

Pro potvrzení či zamítnutí výše uvedených hypotéz byl hlavní cíl práce rozdělen na tři dílčí cíle. První dílčí cíl měl za úkol popsat a otestovat dostupná replikační řešení v návaznosti na jejich funkčnost s prostorovými daty. **Z provedeného testování je výsledkem prvního dílčího cíle potvrzení první hypotézy.** V současné době jsou k dispozici replikační mechanismy v prostorových databázích použitelné pro zabezpečení distribuce prostorových a senzorových dat. Replikace s prostorovými daty je více specifická než replikace klasických datových typů (například číslo či textový řetězec) a mnohdy vyžaduje i specifický přístup. Replikační mechanismy

u PostgreSQL s nadstavbou Slony (asynchronní logická replikace) přináší řadu výhod oproti MySQL. Například Slony dovoluje replikaci jen určitých tabulek (vrstev), replikace je dokončena za mnohem kratší čas (využívá maximální rychlost propojení databázových serverů), který závisí na výkonosti *slave* serveru a počtu lomových bodů v replikovaných datech. Naproti tomu MySQL nezatěžuje tolik CPU a síťové připojení, čas replikace nejvíce ovlivňuje počet replikovaných záznamů, a je zde důležité mít výkonnější *master* server.

Výsledky druhého a třetího dílčího cíle potvrdily druhou hypotézu, že je efektivní provést integraci heterogenních senzorových dat do jednotného datového úložiště. Z ověřených postupů lze efektivně a automatizovaně nastavit celý proces od měření veličin, snížení objemu dat k odeslání, samotné odeslání dat, ošetření hrubých chyb, až po samotné uložení dat do jednotného datového úložiště. Aby byla práce se samotnými daty efektivní, je nutné centralizované datové úložiště distribuovat pomocí replikačních mechanismů, a tím zajistit všem klientům vysokou dostupnost (*high availability*), rozložení zátěže (*load balancing*) a i minimální zálohu dat (i když replikace primárně k zálohování neslouží).

Dalšími výsledky prvního dílčího cíle jsou popisy doporučených forem a struktur používaných distribuovanými prostorovými databázovými systémy pro dosažení distribuovanosti. Dále jsou uvedeny implementační nároky, principy, včetně jednotlivých řešení, které jsou otestovány. Zvláštní zřetel je kladen na replikační a synchronizační mechanismy prostorových databází PostgreSQL a MySQL. Výsledky jsou doprovázeny tabelárními výsledky, grafy a schémata. Následně jsou popsány postupy tvorby distribuované databázové sítě pro konkrétní databázové systémy. Fungování distribuované sítě je popsáno pomocí schémat znázorňujících toky dat v distribuované databázové síti.

Druhý dílčí cíl se zabýval zpracováním senzorových dat produkovaných senzorovými sítěmi komunikujících pomocí technologií Sigfox, LoRaWAN, Zigbee a GPRS. Výsledkem této části jsou zdokumentované metody a formy komunikace mezi senzory, metody odesílání naměřených hodnot pomocí LPWAN sítí (a GPRS) a získávání senzorových dat ze sítě. Závěry z tohoto dílčího cíle byly publikovány v certifikované metodice.

Třetí dílčí cíl si kladl za úkol získaná data očistit a uložit do jednotné prostorové databáze senzorových dat. Získávání dat probíhá pomocí aplikačního middleware, který integruje data z heterogenních senzorových sítí a řeší hrubé chyby v naměřených datech. Middleware je napsán v jazyce Python a běží na cloudové službě Microsoft Azure. Tím, že je middleware napsán v jazyce Python, lze jej provozovat na vlastní nezávislé infrastruktuře. Aby nedošlo k centralizaci dat do fyzicky jediného databázového úložiště, byly využity poznatky z prvního dílčího cíle a byly prakticky otestovány dva možné přístupy vytvoření distribuované databázové sítě. V aplikační rovině byl následně prakticky odzkoušen PostgreSQL nástroj *Notify/-Listen*, který vždy s nově přijatými daty do databáze odeslal notifikaci do QGIS, kde byla data automaticky, bez zásahu uživatele, aktualizována a požadovaná *akce*.

Soudobý technologický vývoj, kdy se desktop aplikace upravují a přesouvají na cloudové služby (např. ArcGIS Online) umožňuje využití nových přístupů ke zpracování dat. Proudová senzorová data, která produkují velké objemy dat, tak lze poměrně snadno převést na webové a cloudové služby, a využít tak jejich přidaný potencionál (platba jen za využitý výkon, vysoká dostupnost, automatické zálohy). Desktop GIS má ovšem stále nezastupitelné místo při zpracovávání prostorových dat a proudová senzorová data budou jistě jedním z dalších běžných zdrojů dat. Funkce *Notify/Listen* rozšiřuje klasické fungování desktop GIS z architektury klient – server na koncept *Publisher – Subscriber,* kdy jsou data pasivně přijímána ze zdroje místo aktivního dotazování, zda existují nějaká nová data.

Pro budoucí zpracování velkého množství senzorových dat by mohly být prozkoumány možnosti aktuálně nabízených cloudových služeb, mimo jiné i Esri ArcGIS Analytics for IoT či Esri ArcGIS GeoEvent Server. Dále pak třeba i nasazení open-source integračních nástrojů typu Apache Kafka na vlastní infrastruktuře s přímým napojením na desktop a WebGIS.

Hlavním přínosem disertační práce je ověřená a popsaná architektura systému, která umožňuje získání senzorových dat z různých senzorů pomocí různých komunikačních technologií, uložení senzorových dat v jednotném datovém distribuovaném úložišti a automatické zpracování a využití prostorových dat pro provádění časoprostorových analýz v rámci desktop GIS v reálném čase.

Díky testování byla ověřena schopnost databázových systémů PostgreSQL a MySQL replikovat prostorová data s následným využitím v distribuované databázové síti. Proudová data ze senzorových sítí jsou poskytována do desktop GIS pomocí PostgreSQL funkce *Nofity/Listen* a tak na straně klienta dochází k tvorbě časoprostorových analýz nad aktuálními daty v reálném čase.

Navržená architektura je vhodná pro systémy zpracovávající senzorová data v GIS v reálném čase. Toto téma je velice aktuální, protože je předpoklad, že takovéto architektury budou v budoucnu stále častěji používány, jak poroste počet senzorů, které generují časoprosotorová data.

8 LITERATURA

[1] Abramova, V.; Bernardino, J.; Furtado, P.: Which NoSQL Database? *Open Journal of Databases (OJDB)*, ročník 1, č. 2, 2014: str. 8.

[2] Anon.: Number Of Database Connections - PostgreSQL wiki. mar 2014. Dostupné z: `https://wiki.postgresql.org/wiki/Number_Of_Database_Connections`

[3] Anon.: Python Developers Survey 2018 Results. 2018. Dostupné z: `https://www.jetbrains.com/research/python-developers-survey-2018/`

[4] Anon: IoT Portál. Technická zpráva, České radiotelekomunikace a.s., Praha, 2019.

[5] Anon: Souhrn smluv o úrovni služeb | Microsoft Azure. feb 2019. Dostupné z: `https://azure.microsoft.com/cs-cz/support/legal/sla/summary/`

[6] Anon.: What is Streaming Data? - Amazon Web Services (AWS). 2019. Dostupné z: `https://aws.amazon.com/streaming-data/`

[7] Anon.: CAT 5, cat 5e, cat6, cat6a, cat7, cat8 Cable Standards. 2020. Dostupné z: `http://www.cablek.com/technical-reference/cat-5---5e--6--6a---7--8-standards`

[8] Anon.: Roadmap - pgpool Wiki. 2020. Dostupné z: `https://www.pgpool.net/mediawiki/index.php/Roadmap`

[9] Anon.: Sigfox Cloud Integration. 2020. Dostupné z: `https://build.sigfox.com/backend-callbacks-and-api`

[10] Boicea, A.; Radulescu, F.; Agapin, L. I.: MongoDB vs Oracle - Database comparison. *Proceedings - 3rd International Conference on Emerging Intelligent Data and Web Technologies, EIDWT 2012*, 2012: s. 330–335, doi: 10.1109/EIDWT.2012.32.

[11] Buttazzo, G.: Real-time operating systems: Problems and novel solutions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, ročník 2469, 2002: s. 37–51, ISSN 16113349.

[12] Campoli, F.: pg_chameleon. 2020. Dostupné z: `https://pgchameleon.org/`

[13] Castell, N.; Dauge, F. R.; Schneider, P.; aj.: Can commercial low-cost sensor platforms contribute to air quality monitoring and exposure estimates? *Environment International*, ročník 99, feb 2017: s. 293–302, ISSN 01604120, doi:10.1016/j.envint.2016.12.007.

[14] Charvát, K.; Kocáb, M.; Konečný, M.; aj.: *Geografická data v informační společnosti*. Výzkumný ústav geodetický, topografický a kartografický, v.v.i., 2007, ISBN 978-80-85881-28-8, 280 s.

[15] Dolgikh, M.: *Vysoká dostupnost v relačních databázových systémech*. Diplomová práce, Masarykova univerzita, Brno, 2014.

[16] Ellis, B.: Real-time Analytics: Techniques to Analyze and Visualize Streaming Data. 2014: str. 435.

[17] Filippovová, J.; Pohanka, T.: Environmental Assessment of Central European Floodplain Forests: A Case Study from the Morava River Alluvium. *Polish Journal of Environmental Studies*, ročník 28, č. 6, sep 2019: s. 4511–4517, ISSN 1230-1485, doi:10.15244/pjoes/95041.

[18] Gartner: Gartner Says 6.4 Billion Connected "Things"Will Be in Use in 2016, Up 30 Percent From 2015. 2015.
Dostupné z: `http://www.gartner.com/newsroom/id/3165317`

[19] Hejlová, V.: *Experimentální bezdrátová senzorová síť pro monitoring znečištění ovzduší ve středu města Olomouce (E-BOSS)*. Dizertační práce, Univerzita Palackého v Olomouci, 2017.

[20] Hohpe, G.; Woolf, B.: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003, ISBN 0321200683.

[21] Kabakus, A. T.; Kara, R.: A performance evaluation of in-memory databases. *Journal of King Saud University - Computer and Information Sciences*, ročník 29, č. 4, 2017: s. 520–525, ISSN 22131248, doi:10.1016/j.jksuci.2016.06.007.

[22] Kulkarni, A.: Time-series data simplified | Timescale. 2020.
Dostupné z: `https://www.timescale.com/`

[23] Lake, P.; Crowther, P.: *Concise Guide to Databases*. Undergraduate Topics in Computer Science, London: Springer London, 2013, ISBN 978-1-4471-5600-0, doi:10.1007/978-1-4471-5601-7.

[24] Longley, Paul. Goldchild, Mike. Maguire, David. Rhind, D.: *Geographic Information Systems & Science*. Třetí vydání, 2011, ISBN 9780470721445, 539 s.

[25] Mazilu, M. C.: Database Replication. *Database Syst. J.*, ročník I, č. 2, 2010: s. 33–38, ISSN 21508097.

[26] Mekki, K.; Bajic, E.; Chaxel, F.; aj.: A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express*, ročník 5, č. 1, 2018: s. 1–7, ISSN 24059595, doi:10.1016/j.icte.2017.12.005.

[27] Nétek, R.; Pohanka, T.: Development of android map application for field collection of agriculture data. *Listy Cukrovarnicke a Reparske*, ročník 135, č. 2, 2019: s. 67–70, ISSN 18059708.

[28] Ondruška, M.: Architektura informačního systému z pohledu integrace v kontextu podnikové integrace. *Systémová Integr.*, ročník 4, 2009: s. 143–158.

[29] Özsu, M. T.; Valduriez, P.: *Principles of distributed database systems.* 2011, ISBN 9781441988331, 1–845 s., doi:10.1007/978-1-4419-8834-8, `arXiv:1011.1669v3`.

[30] Pechanec, V.; Machar, I.; Pohanka, T.; aj.: Effectiveness of Natura 2000 system for habitat types protection: A case study from the Czech Republic. *Nature Conservation*, ročník 24, jan 2018: s. 21–41, ISSN 1314-3301, doi:10.3897/natureconservation.24.21608.

[31] Pechanec, V.; Pohanka, T.; Julina, V.; aj.: Integrace obrazových materiálů s daty ze senzorové sítě. Technická zpráva, Univerzita Palackého v Olomouci, Olomouc, 2017.
Dostupné z: `http://gislib.upol.cz/moseso/file/metodikaTA04020888-integrace_SS.pdf`

[32] Pechanec, V.; Pohanka, T.; Kilianová, H.; aj.: Variability of soil carbon on land borderlines. *Listy Cukrovarnicke a Reparske*, ročník 135, č. 4, 2019: s. 148–151, ISSN 18059708.

[33] Pohanka, T.; Kilianová, H.; Pechanec, V.: Space-temporal analysis of soil organic characteristics on the ecotone border. *Fresenius Environmental Bulletin*, ročník 28, č. 4A/2019, 2019: s. 3141–3146.

[34] Pohanka, T.; Pechanec, V.; Hejlová, V.: Python Web Server for Sensor Data Visualization. In *16th International Multidisciplinary Scientific Conference SGEM2016, Book 2*, ročník 1, jun 2016, ISBN 978-619-7105-58-2, ISSN 1314-2704, s. 803–810, doi:10.5593/sgem2016B21.

[35] Pohanka, T.; Pechanec, V.; Solanska, M.: Synchronization and replication of geodata in the ESRI platform. *International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, SGEM*, ročník 1, č. 2, 2015: s. 837–843, ISSN 13142704, doi:10.5593/sgem2015/b21/s8.107.

[36] Pour, J.; Gála, L.; Zuzana, Š.: *Podniková Informatika*. Grada Publishing, a.s., 2009, ISBN 978-80-247-2615-1, 496 s.

[37] Reinsel, D.; Gantz, J.; Rydning, J.: Data Age 2025: The Digitization of the World From Edge to Core. *International Data Corporation*, , č. November, 2018: str. 28.

[38] Rieke, M.; Bigagli, L.; Herle, S.; aj.: Geospatial IoT - the need for event-driven architectures in contemporary spatial data infrastructures. *ISPRS International Journal of Geo-Information*, ročník 7, č. 10, 2018: s. 1–29, ISSN 22209964, doi:10.3390/ijgi7100385.

[39] Shirer, M.; MacGillivray, C.: The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast. jul 2019.
Dostupné z: `https://www.idc.com/getdoc.jsp?containerId=prUS45213219`

[40] Singer, S.: Slony-I. 2020.
Dostupné z: `http://www.slony.info/`

[41] Solanská, M.: *Synchronizace a replikace geodat v prostředí ESRI platformy.* Magisterská práce, Palackého Univerzita v Olomouci, 2014.

[42] Sutherland, J.; van den Heuvel, W.-J.: Enterprise application integration and complex adaptive systems. *Communications of the ACM*, ročník 45, č. 10, 2002, ISSN 00010782, doi:10.1145/570907.570932.

[43] Trnová, L.: *Hodnocení mechanismů replikace vybraných databázových systémů.* Bakalářská práce, Univerzita Palackého v Olomouci, 2018.

[44] Urbano, R.: *Oracle Database Advanced Replication*, ročník 1. Oracle, 2015, 222 s.

[45] Vrublova, K.; Pohanka, T.: Is Environmental Conservation of European Beech-Dominated Forests Efficient? *Fresenius Environmental Bulletin*, ročník 28, č. 2A, 2019: s. 1218–1223, ISSN 1018-4619.

[46] Wingerath, W.; Ritter, N.; Gessert, F.: *Real-Time & Stream Data Management.* SpringerBriefs in Computer Science, Cham: Springer International Publishing, 2019, ISBN 978-3-030-10554-9, doi:10.1007/978-3-030-10555-6.

ODBORNÝ ŽIVOTOPIS AUTORA

Osobní údaje

Jméno	Tomáš POHANKA
Bydliště:	Nám. Jana Zajíce 2958/13, 787 01 Šumperk
E-mail:	tomas.pohanka@upol.cz
Telefon:	+420 775 13 13 94
Narozen:	14. 2. 1989, Náchod

Vzdělání

2014-dosud	Univerzita Palackého v Olomouci, Přírodovědecká fakulta obor: Geoinformatika a kartografie (Ph.D.) Téma: Distribuované geodatabáze senzorových dat – základ pro integraci a analýzu
2011-2014	Univerzita Palackého v Olomouci, Přírodovědecká fakulta obor: Geoinformatika (Mgr.) Obhájená práce: Fuzzy modely pro efektivní řízení křižovatky – případová studie ve městě Trnava
2008-2011	Univerzita Palackého v Olomouci, Přírodovědecká fakulta obor: Geoinformatika a geografie (Bc.) Obhájená práce: Evidence územních plánů pro Olomoucký kraj

Absolvované stáže

Akad. rok	Instituce	Termín (počet týdnů)
2014/15	Technische Universität Dresden	4. 4. – 22. 5. 2015 (7 týdnů)
2016/17	Debrecen University	22. 5. – 26. 6. 2017 (5 týdnů)

Praxe

2018-dosud	Univerzita Palackého v Olomouci, Katedra Geoinformatiky Technik informačních systémů

Absolvované předměty

Povinné	Splněno
Management vědy a výzkumu	9. 12. 2014
Anglický jazyk pro doktorské studium	22. 6. 2015
Vědecko-výzkumná stáž	8. 8. 2017
Povinné oborové	
Objektově orientované technologie	30. 8. 2016
Vývoj softwarových prostředků pro open-source GIS	17. 8. 2016
Servisně orientovaná architektura v geoinformatice	17. 8. 2017
Systémy pro podporu prostorového rozhodování	30. 11. 2017
Povinně volitelné publikační činnost	
Hlavní autor publikace v časopise s IF	17. 4. 2019
Spoluautorství v časopise s IF	17. 6. 2019
Spoluautorství v časopise s IF	20. 2. 2018
Povinně volitelné vědecko-výzkumná a pedagogická činnost	
Výuka odborného předmětu na UP	25. 6. 2015
Projektová činnost	25. 6. 2015
Projektová činnost	18. 8. 2016
Projektová činnost	31. 7. 2017
Ústní prezentace na mezinárodní konferenci	31. 7. 2017
Prezentace na konferenci (poster)	6. 12. 2017
Teze k disertační práci	8. 2. 2018
Volitelné – ostatní	
Oponování závěrečných prací	25. 6. 2015
Popularizační aktivity na UP	18. 8. 2016
Popularizační aktivity na UP	4. 5. 2017
Vedení bakalářských prací	25. 6. 2018

Podíl na výuce

Předmět	Název
KGI/GISON	GIS Online
KGI/GIT	Geoinformační technologie
KGI/DYWE	Dynamický web
KGI/FREOS	Free a Open Source
KGI/PRG1	Programování 1

Zapojení do projektů

Výzkum a aplikace metod geoinformatiky pro řešení prostorových jevů reálného světa

- IGA UP projekt IGA_PrF_2019_014 (pracovník projektu)

Významné stromy – živé symboly národní a kulturní identity :

- NAKI: DG18P02OVV027 (NAKI II) (pracovník projektu)

Systém automatizovaného monitorování a modelování znečištění podzemních vod z nebodových průmyslových zdrojů (SAMMWAP):

- TAČR: TH03030023 (pracovník projektu)

Inovativní metody hodnocení a pokročilé analýzy prostorově založených systémů

- IGA UP projekt IGA_PrF_2018_028 (pracovník projektu)

Cloudová platforma pro integraci a vizualizaci různých typů geodat

- IGA UP projekt IGA_PrF_2017_024 (pracovník projektu)

Multiplatformní systém pro identifikaci a správu komářích líhnišť

- Inovační voucher Olomouckého kraje – Geocentrum, s.r.o, CZ.01.1.02/0.0/0.0/16_045/0008076 (pracovník projektu)

Pokročilý monitoring, prostorové analýzy a vizualizace městské krajiny

- IGA UP projekt IGA_PrF_2016_008 (pracovník projektu)

Bezkontaktní monitorování a časoprostorové modelování variability vybraných diferenciačních vlastností půdy

- TAČR Alfa: TA04020888 (pracovník projektu)

Pokročilá integrace senzorových sítí a bezkontaktního monitoringu krajiny v oblasti precizního zemědělství

- IGA UP projekt IGA_PrF_2015_012 (pracovník projektu)

Budování výzkumně-vzdělávacího týmu v oblasti modelování přírodních jevů a využití geoinformačních systémů, s vazbou na zapojení do mezinárodních sítí a programů (StatGISTeam)

- OPVK, CZ.1.07/2.3.00/20.0170 (pracovník projektu)

Publikace

Odborný článek [J]

NÉTEK, R., POHANKA, T. Vývoj Android mapové aplikace pro terénní sběr zemědělských dat. In: Listy cukrovarnické a řepařské, 2019

PECHANEC, V., POHANKA, T., KILIANOVÁ, H., FIŠEROVÁ, E., ŘÍMALOVÁ, V. Variabilita půdního uhlíku na hranicích pozemků. In: Listy cukrovarnické a řepařské, 2019

FILIPPOVOVÁ, J., POHANKA, T. Environmental Assessment of Central European Floodplain Forests (A Case Study from the Morava River Alluvium) In: Polish Journal of Environmental Studies, 2019

VRUBLOVÁ, K., POHANKA, T. Is environmental conservation of European beech-dominated forest efficient? In: Fresenius Environmental Bulletin, 2019

PECHANEC, V., MACHAR, I., POHANKA, T., OPRŠAL, Z., PETROVIČ, F., ŠVAJDA, J., ŠÁLEK, L., CHOBOT, K., FILIPPOVOVÁ, J., CUDLÍN, P., and MÁLKOVÁ, J. Effectiveness of Natura 2000 system for habitat types protection: A case study from the Czech Republic. In: Nature Conservation, 24, 21–41. 2018

Publikace v recenzovaném konferenčním sborníku v databázi WoS, Scopus (D)

NÉTEK, R., POHANKA, T., VOŽENÍLEK, V. Implementation of Geospatial Web Services for Precise Farming - Case study on Responsive Map Client. In: ICGDA2018: Conference Proceedings. Prague, 2018.

POHANKA Tomáš, Vilém PECHANEC, Alexander MRÁZ, Helena Kiliánová a Antonín BENC, The integration of sensor data into gis on the example of spectral reflectance sensor. In: SGEM2017: Conference Proceedings. Sofia, Bulgaria: STEF92 Technology Ltd., 2017.

POHANKA Tomáš, Vilém PECHANEC a Marek BALUN, Web-GIS application for analyses of the sensors data on the web. In: SGEM2017: Conference Proceedings. Sofia, Bulgaria: STEF92 Technology Ltd., 2017.

VONDRÁKOVA Alena, Rostislav NÉTEK, Tomáš POHANKA, Tomáš POUR, Michal MALACKA, Legislative aspects of spatial data publication. In: SGEM2017: Conference Proceedings. Sofia, Bulgaria: STEF92 Technology Ltd., 2017.

HEJLOVÁ Vendula, POHANKA Tomáš, Configuration of Wireless Sensor Network in Olomouc. GIS Ostrava 2017

POHANKA, Tomáš, Vilém PECHANEC a Vendula HEJLOVA, Python web server for sensor data visualization. In: SGEM2016: Conference Proceedings. Sofia, Bulgaria: STEF92 Technology Ltd., 2016.

HEJLOVÁ, V., POHANKA, T., BUTAZZO, W., PECHANEC, V., NWAUOGU, C. Communication Distance of Jennic Wireless Nodes in the Small Area. In: SGEM2015: Conference Proceedings. Sofia, Bulgaria: STEF92 Technology Ltd., 2015.

POHANKA, Tomáš, Vilém PECHANEC a Markéta SOLANSKÁ. Synchronization and replication of geodata in the esri platform. In: SGEM2015: Conference Proceedings. Sofia, Bulgaria: STEF92 Technology Ltd., 2015.

Certifikovaná metodika

PECHANEC,V., POHANKA, T., JULINA, V., KILIANOVÁ, H., ULIČNÍK, B. Integrace obrazových materiálů s daty ze senzorové sítě. Certifikovaná metodika. Ústřední kontrolní a zkušební ústav zemědělský číslo certifikace: UKZUZ 020993/2017.

Software (R)

PECHANEC, V., POHANKA, T. AGISED - Integrace obrazových materiálů s daty ze senzorové sítě pro zemědělství. Software. 2017. TA04020888-2015V002

PECHANEC,V., POHANKA, T. SoilVar - program pro modelování časo-prostorové variability půdních vlastností. Software. 2017

PECHANEC, V,. POHANKA, T., KILIÁNOVÁ, H. Porovnání půdních vlastností vybraných ekosystémů. Software. 2015. TA04020888-2015V002

Specializovaná mapa (Mapa)

PECHANEC, V., VONDRÁKOVA, A., VRÁNOVÁ, V., KILIANOVÁ, H., POHANKA, T. Vybrané půdní charakteristiky v lokalitách Proklest a Rudice. Mapa, 2017

MIŘIJOVSKÝ, J., BRUS, J., HEJLOVÁ, V., POHANKA, T. Hodnocení vývoje zemědělských plodin na základě dat DPZ. Mapa, 2015. IGA_PrF_2015_012/2

Ostatní publikace

POHANKA, Tomáš, Distributed spatial database systems for sensor data. In: VIII. GIS Conference and Exhibition: Conference Proceedings, Debrecen, Hungary, 2017. ISBN 978-963-318-638-1

POHANKA, Tomáš, Aplikace pro vizualizaci senzorových dat, Sborník rozšířených abstraktů. Digitální technologie v Geoinformatice, kartografii a dálkovém průzkumu Země, 25. 10. 2016, s. 25-26. ISBN 978-80-01-06019-3.

POHANKA, Tomáš. Vizualizace senzorových dat na webu pomocí Python, Sborník abstraktů. ISAF & Geomatics in Projects & Plan4all Joint Conference, 2016. ISBN 978- 80-263-1103-4

POHANKA, Tomáš. Fuzzy models for efficient traffic management. In: Second StatGIS Conference: proceedings : 10th - 13st november 2014, Olomouc, Czech Republic : StatGIS team. Olomouc: Palacký University, 2013, s. 56-60. ISBN 978-80-244-4273-0

POHANKA, Tomáš. Souřadnicové systémy online. In: Second StatGIS Conference: proceedings : 10th - 13st november 2014, Olomouc, Czech Republic : StatGIS team. Olomouc: Palacký University, 2013, s. 124-126. ISBN 978-80-244-4273-0

Přehled aktivní účasti na konferencích

GIS Ostrava 2019, 20. 3. – 22. 3. 2019, Ostrava, příspěvěk: Kontinuální monitorování a modelování nebodového znečištění podzemních vod.

SGEM 2017, 26. 7 – 6. 8. 2017, Albena, Bulharsko, příspěvek: Web-GIS application for analyses of the sensors data on the web.

SGEM 2017, 26. 7 – 6. 8. 2017, Albena, Bulharsko, příspěvek: The integration of sensor data into gis on the example of spectral reflectance sensor.

VIII. GIS Conference, 25. 6 - 26. 6. 2017, Debrecen, Hungary, příspěvek: Distributed spatial database systems for sensor data.

Digitální technologie v GIS, kartografii a DPZ, 25. 10. 2016, Praha, příspěvek: Aplikace pro vizualizaci senzorových dat.

JOINT 2016, 4. – 6. 10. 2016, Plzeň, příspěvek: Vizualizace senzorových dat na webu pomocí Python.

2nd StatGIS Conference, 18. – 21. 11. 2014, Olomouc, příspěvek: Souřadnicové systémy online.

2nd StatGIS Conference, 18. – 21. 11. 2014, Olomouc, příspěvek: Fuzzy models for efficient traffic management.

ANNOTATION

The main objective of this doctoral thesis was analysing and verification of system architecture, which enable to collect, store, process and share large-scale data generated by modern, geographically distributed wireless sensor networks. Architecture is using distributed spatial database systems for creation of spatiotemporal analysis.

Thesis present author's approach for building wireless sensor networks using LPWAN (Low Power Wide Area Networks) technologies, preprocessing gained data for future usage in GIS and creating distributed spatial database site to increase data availability. The motivation for the doctoral thesis was a demand to make streaming sensor data accessible in desktop GIS. Desktop GIS is mostly used for creating advanced spatiotemporal analysis.

The thesis should answer two primary research questions:

1. Are nowadays available replication mechanisms for spatial databases secure in terms of distributing spatial and sensor data?

2. Is it efficient to integrate heterogeneous sensor data into unified data storage?

The main objective of doctoral thesis was split into three sub-objectives in order to confirm or reject the hypothesis. The first sub-objective aims to describe and test available database replication solutions which provide functionality over spatial data. The second sub-objective aims to sensor data processing produced by LPWAN sensor networks Sigfox, LoRaWAN, Zigbee and GPRS (not low powered). The third sub-objective aims to clear the gained data from gross errors and storing it into the unified spatial database. To avoid data centralisation, was the database distributed by replication mechanisms.

Keywords replication; spatial database; sensor; communication; middleware
Number of standard page: 45

SUMMARY

The main objective of this doctoral thesis was analysis, design and verification of system architecture, which enables to collect, store, process and share large-scale data generated by modern, geographically distributed wireless sensor networks. Architecture is using distributed spatial database system for the creation of spatiotemporal analyses within GIS in real-time.

The thesis presents author's approach for building wireless sensor networks using LPWAN (Low Power Wide Area Networks) technologies, preprocessing of gathered data for real-time processing within GIS and creating distributed spatial database site to increase data availability. The motivation for the doctoral thesis was a demand to make streaming sensor data accessible in desktop GIS in real-time. Desktop GIS is mostly used for creating advanced spatiotemporal analysis.

The thesis ought to answer two primary research questions:

1. Are there nowadays available replication mechanisms of spatial databases suitable for distribution of spatial and sensor data?

2. Is it efficient to integrate heterogeneous sensor data into unified data storage?

The main objective of the doctoral thesis was split into three sub-objectives to confirm or reject the hypothesis. The first sub-objective aims to describe and test available database replication solutions which provide replication functionality over spatial data. The results from performed testing of the replication mechanisms confirm the first hypothesis. There are nowadays available replication mechanisms of spatial databases for distribution of spatial and sensor data. Spatial data replication is more specific and requires a specific approach than a replication of common data types. PostgreSQL with Slony-I extension and MySQL with build-in replication were chosen for testing. Slony-I allows replication of one or more tables among PostgreSQL servers and replication is performed in a shorter time than in MySQL. Slony-I replication time depends on the power of a slave server and quantity of edge points in a replicated data. MySQL replication does not require as much CPU power as PostgreSQL and is less demanding on network bandwidth. MySQL replication time is mainly influenced by the number of records and CPU power of a master server.

The results of second and third sub-objective confirm the second hypothesis that integration of heterogeneous sensor data into unified data storage is indeed effective. It is possible to take very efficient and automated steps during the whole process starting from measuring of variables, reducing data size for transfer from the sensor, data transfer itself, solving rough errors in data and ending with storing sensor data into unified data storage. To gain even more efficiency, it is necessary to distribute data storage by replication mechanisms. Distributed storage provides high availability to all clients and load balancing.

Additional results from the first sub-objective are descriptions of recommended forms and structures used for distributed spatial database systems to be distributable. Implementation requirements, principles and solutions for deployment of distributed databases systems are also mentioned.

The second sub-objective aims to processing of sensor data produced by LPWAN sensor networks Sigfox, LoRaWAN, Zigbee and GPRS (not low powered). Results from this sub-objective are well-documented methods and forms of communication among sensors and gates, as well as methods of transmitting measured data by LPWAN and obtaining these data from LPWAN network. Results from this sub-objective are published in certificated methodology.

The third sub-objective aims to purge obtained data from gross errors and storing it into the unified spatial database. Data acquisition from LPWAN networks was performed by application middleware, which integrates data from heterogeneous sensor networks. Middleware is written in Python, and is deployed on the Microsoft Azure cloud platform. The universality of Python scripting language provides a the possibility of running the middleware on any server or computer.

The architecture was verified also at the application level by testing of PostgreSQL's tool Notify/Listen. This tool is used for notifying QGIS about newly added sensor data into a database. QGIS subsequently automatically performs an action or spatiotemporal analysis over newly received data.

The main contribution of this doctoral thesis is the verified and well-described architecture that allows acquisition of sensor data from different sensors that communicate by different communication technologies. The architecture further allows efficiently integrate and store these heterogeneous data into a unified distributed spatial database that allows performing spatiotemporal analysis by processing sensor data within QGIS.

Efficient spatial data replication of PostgreSQL and MySQL database systems were verified by testing. Spatial data were consequently available within a distributed database site. Streaming sensor data are provided into desktop GIS using PostgreSQL tool "Notify" and Listen function in QGIS for automated creation of advanced spatiotemporal analysis over actual data in real-time.

Architecture designed in this doctoral thesis is suitable for systems that process sensor data within GIS in real-time. This topic is very actual, because there is a premise, that such architectures will be more and more deployed, due to ever-increasing amount of sensors that generate spatiotemporal data.

DISTRIBUOVANÉ GEODATABÁZE SENZOROVÝCH DAT
ZÁKLAD PRO INTEGRACI A ANALÝZU

DISTRIBUTED SENSOR GEODATABASE
PLATFORM FOR INTEGRATION AND ANALYSIS

AUTOREFERÁT DISERTAČNÍ PRÁCE
Phd THESIS SUMMARY

Předkladatel / Submitter:
Mgr. Tomáš Pohanka

Určeno pro studenty, partnerská akademická pracoviště a odbornou veřejnost.

Výkonný redaktor: Mgr. Miriam Delongová
Odpovědná redaktorka: Mgr. Lucie Loutocká
Technická redakce: Mgr. Tomáš Pohanka

Publikace neprošla redakční jazykovou úpravou.

Vydala a vytiskla Univerzita Palackého v Olomouci
Křížkovského 8, 771 47 Olomouc
www.vydavatelstvi.upol.cz
www.e-shop.upol.cz
vup@upol.cz

1. vydání
Olomouc 2020
Edice GEOINFO-CARTO-THESIS, svazek XVIII.

ISSN 1805-7500
ISBN 978-80-244-5684-3

Neprodejná publikace